RAPORT ȘTIINȚIFIC ȘI TEHNIC

Proiect 680PED/2022 – "PROFET"
Poziționarea precisă pe orbită a unui satelit folosind tehnici de fuziune de date și estimare bazată pe model

Raportare Etapa 2: 13 decembrie 2023

Cuprins

1 Rezumatul etapei 2

Primul obiectiv al acestei etape a fost validarea modelului de satelit (A-2.1). Apoi, s-a făcut studiul algoritmilor de estimare (A-2.2), s-a particularizat adaptarea acestora pentru aplicaţii aerospaţiale (A-2.3) şi s-a realizat programarea acestora în limbaje avansate (A-2.4). Ulterior, s-au definit specificaţiile hardware (A-2.5), s-a proiectat circuitul electronic (A-2.6) şi s-a realizat integrarea hardware/software a sistemului (A-2.7).

Toate aceste activităţi s-a concretizat prin livrabilele:

D-2.2 *model obţinut post-validare;*
D-3.1 *raport tehnic privind estimarea;*
D-3.2 *bibliotecă de algoritmi de estimare;*
D-4.1 *raport tehnic privind modulul electronic;*
D-4.2 *placa hardware obţinută*;
D-6.1 *diseminarea rezultatelor în mediul academic.*

Cele cinci rapoarte tehnice de la livrabilele D-2.2, D-3.1, D-3.2, D-4.1 şi D-4.2 au fost integrate în acest material pentru realizarea unui document unitar, care să ilustreze într-o manieră coerentă desfăşurarea activităţilor pentru urmărirea obiectivelor propuse pentru Etapa 2 pe 2023 a proiectului de cercetare PROFET.

Câte o placă de dezvoltare pentru ultima etapă a proiectului pe 2024 se regăseşte la fiecare dintre cele două entităţi academice, UPB (UNSTPB) şi UB, iar bibliotecile de algoritmi se pot obţine cu parolă de la link-urile din S-2.4.

Articolele de conferinţă acceptate precum şi cele de revistă acceptate sau în curs de evaluare, care conţin mesaj de Acknowledgment pentru finanţarea primită prin proiectul PROFET, conform activităţii A-2.8 care a avut ca rezultat livrabilul D-6.1, sunt listate în S-2.8.

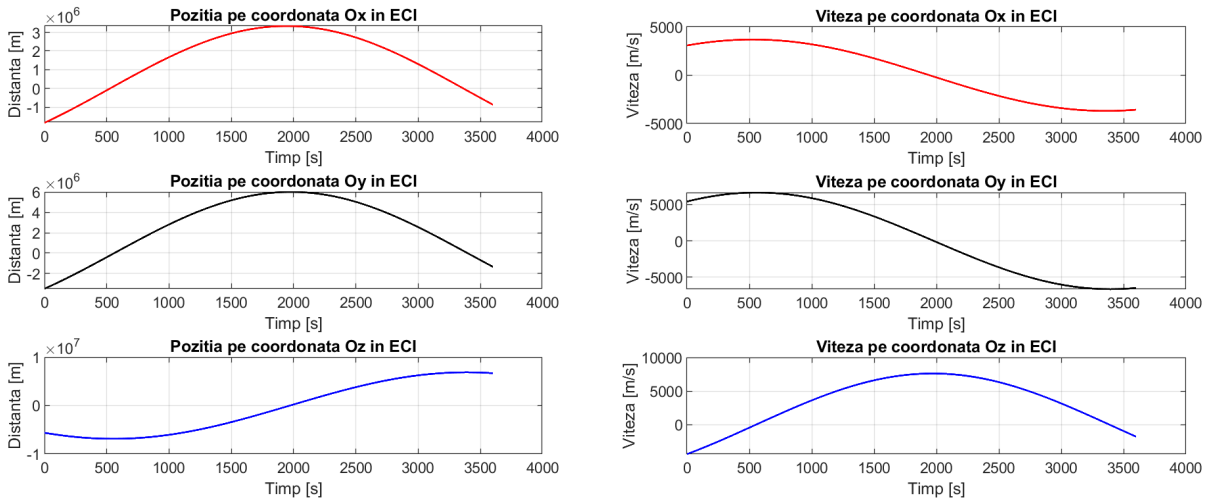
2 Raportul in extenso pe 2023

2.1 Validarea modelului de satelit

În cadrul raportului precedent am prezentat mai multe metode de modelare; dintre metodele prezentate ne vom axa în cele ce urmează pe validarea modelului ce are la bază setul de perturbaţii ce acţionează asupra satelitului cu ajutorul unui propagator de stare. În cele ce urmează vom reaminti care este setul de perturbaţii pe care le putem include în dinamica satelitului, vom prezenta performanţele propagatorului de stare în estimarea stării, vom compara rezultatele obţinute cu ajutorul propagatorului de stare cu rezultatele obţinute cu ajutorul utilitarului Systems Tool Kit (STK), în cele din urmă vom realiza o analiză a timpului de execuţie şi a radicalului erorii medii pătratice în funcţie de perturbaţiile considerate în etapa de propagare a stării.

Perturbaţiile considerate Vom considera următorul set de perturbaţii ce acţionează asupra satelitului GRACE-FO: atracţia gravitaţională a Soarelui, a Lunii şi a planetelor din Sistemul Solar, presiunea exercitată de radiaţia solară, presiunea exercitată de atmosferă, dar şi efectele cauzate de geopotenţialul Pământului la care am ţinut cont şi de influenţa mareelor Pământului şi ale oceanelor.

Propagatorul de stare Pentru propagarea stării am utilizat un propagator de stare open-source pus la dispoziţie de către *Mathworks*. Astfel prin includerea setului de perturbaţii menţionat anterior şi alegerea unui ordin egal cu 70 pentru influenţele geopotenţialului obţinem alura din Figura 1 pentru poziţia şi viteza satelitului. Pentru iniţializarea propagatorului de stare utilizăm poziţia satelitului din 31/12/2021 de la ora 23:59:47.



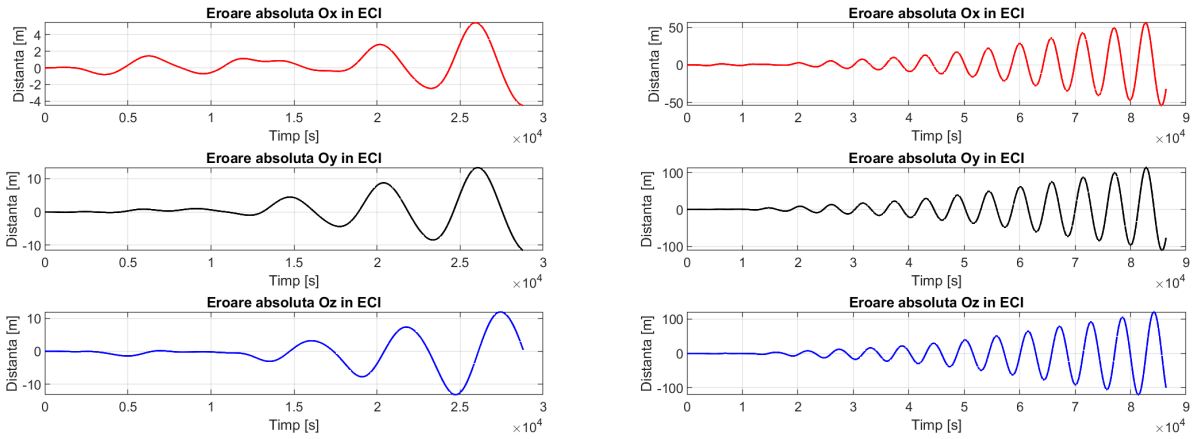
(a) Estimare poziţie satelit GRACE-FO

(b) Estimare viteză satelit GRACE-FO

Figura 1: Estimare vector de stare pentru satelitul GRACE-FO pe durată de o oră

Performanţele propagatorului de stare Pentru validarea propagatorului vom utiliza baza de date pusă la dispoziţie de NASA împreună cu German Research Centre for Geosciences (GFZ) [1], ce

conţine date despre traiectoria satelitului. Vom reface propagarea stării începând de la data menţionată anterior şi vom prezenta eroarea dintre rezultatele propagării şi traiectoria adevarată a satelitului. Observăm în Figura 2 cum erorile de estimare a poziţiei cu ajutorul unui propagator de stare sunt extrem de mici. Totuşi pentru aceste rezultate am utilizat un ordin ridicat pentru coeficienţii geopotenţialului, fapt ce ne conduce la un timp de rulare ridicat. Astfel în etapele următoare vom prezenta rezultate ce au la bază un ordin $n = m = 21$ pentru termenii de geopotenţial.



(a) Estimare poziţie satelit GRACE-FO pe o durată de 8 ore

(b) Estimare poziţie satelit GRACE-FO pe o durată de 24 de ore

Figura 2: Estimare poziţie satelit GRACE-FO

Comparaţie cu STK În continuare vom compara performanţele propagatorului open-source cu cele din Systems Tool Kit cunoscut sub acronimul de STK; STK este un produs de la Ansys ce permite realizarea analizelor complexe asupra sistemelor. Îl vom utiliza pentru simularea dinamicii orbitale a satelitului GRACE-FO şi mai ales pentru testarea limitelor propagatorului prezentat anterior. Pentru că aplicaţia STK de la Ansys nu este open-source şi nu am avut acces la o licenţă pentru a genera un set de date în aceleaşi condiţii ca cele pentru propagatorul open-source, vom modifica parametrii propagatorului open-source pentru a utiliza aceiaşi parametri ca şi cei utilizaţi de STK la generarea estimării orbitei satelitului GRACE-FO. Pentru consistenţa comparaţiei dintre cele două propagatoare am ales următorul set de perturbaţii pentru propagatorul open-source: atracţia gravitaţională a Soarelui şi a Lunii, perturbaţia exercitată de radiaţia solară, presiunea exercitată de atmosferă, dar şi efectele geopotenţialului Pământului păstrând coeficienţii până la ordinul 21.
Obervăm că erorile de estimare a vectorului de stare sunt similare pentru o perioadă de 2 ore (durata efectuării unei orbite complete în jurul Pământului pentru satelitul GRACE-FO fiind de aproximativ 95 de minute). Vom analiza în continuare norma 2 a erorii de estimare pe cele două ore şi eroarea relativă maximă de poziţionare.

	x_{ECI}	y_{ECI}	z_{ECI}
Norma 2 STK	279.56	492.51	260.38
Norma 2 OS	321.29	576.73	304.10
Eroare de estimare STK [%]	3.15×10^{-4}	2.39×10^{-3}	1.09×10^{-4}
Eroare de estimare OS [%]	4.65×10^{-4}	1.71×10^{-3}	2.57×10^{-4}

Tabela 1: Comparaţie erori STK vs. propagator open-source

Obeservăm că erorile pe întregul interval de estimare a stării sunt cu aproximativ 18% mai mari în cazul estimatorului open-source pentru acelaşi configuraţie de estimare. Totuşi uitându-ne la erorile relative de estimare observăm că acestea sunt extrem de mici, având acelaşi ordin de mărime pentru

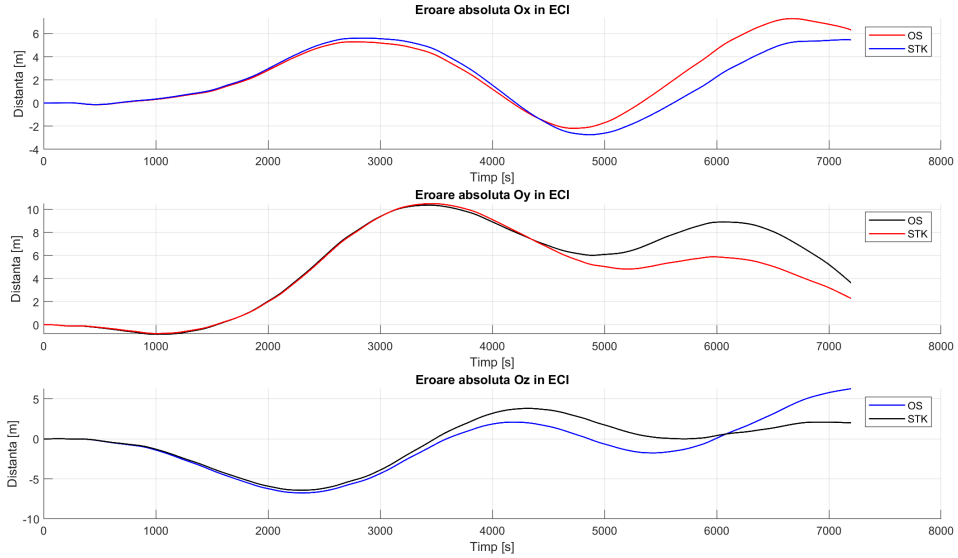


Figura 3: Eroarea de estimare a poziţiei pentru satelitul GRACE-FO cu cele două propagatoare

fiecare dintre cele două propagatoare pe fiecare axă în parte. Putem astfel să concluzionăm că propagatorul open-source este foarte bun şi prin comparaţie cu un utilitar specializat în simularea dinamicii orbitale.

Alegerea setului de perturbaţii Vom analiza influenţa perturbaţiilor asupra timpilor de rulare, dar şi a erorilor medii pătratice. Pentru realizarea analizei vom adăuga secvenţial perturbaţii; vom începe cu perturbaţiile cauzate de geopotenţialul terestru pentru care vom considera un ordin $n = m = 21$. La această perturbaţie vom adăuga pe rând perturbaţiile cauzate de atracţia gravitaţională a Soarelui, a Lunii, a frecării atmosferice şi a radiaţiei solare. Graficul aferent acestei etape de simulare se regăseşte în Figura 4 şi reprezintă rezultatele obţinute pentru propagarea a aproximativ două orbite ale GRACE-FO în jurul Pământului din luna Decembrie a anului 2021.

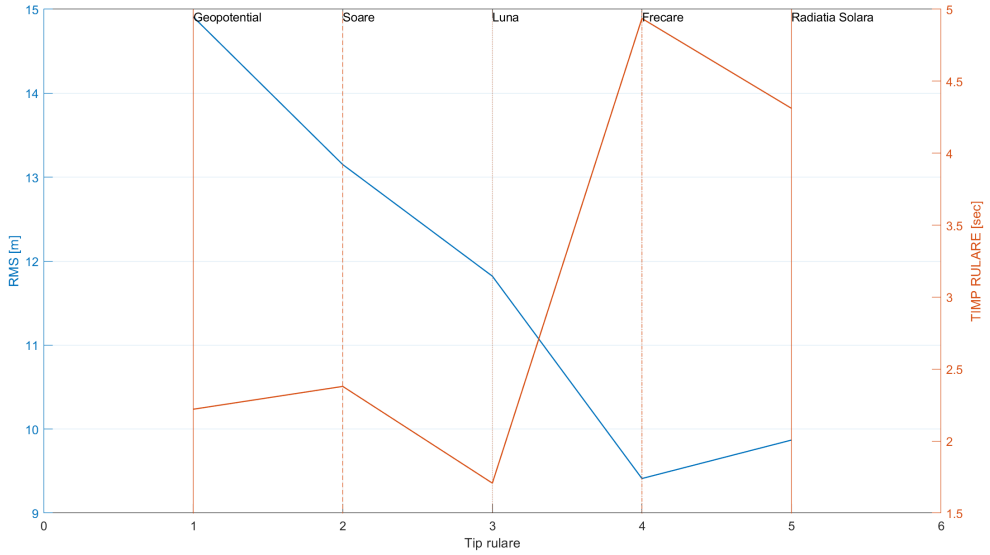


Figura 4: Radicalul erorii medii pătratice vs. timpul mediu de rulare pentru diferite configuraţii de perturbaţii - pentru orbita GRACE-FO din 31 Decembrie 2021

Pentru calculul timpilor medii de rulare am realizat un număr de 100 de propagări pentru fiecare configuraţie de perturbaţii menţionată anterior; din datele obţinute din simulare am eliminat erorile grosiere. Am repetat simulările în aceleaşi condiţii pentru a verifica reproductibilitatea rezultatelor obţinute; astfel vom verifica eroarea absolută dintre timpii medii de rulare pentru cele 2 iteraţii de simulare efectuate; vedeţi Figura 5.

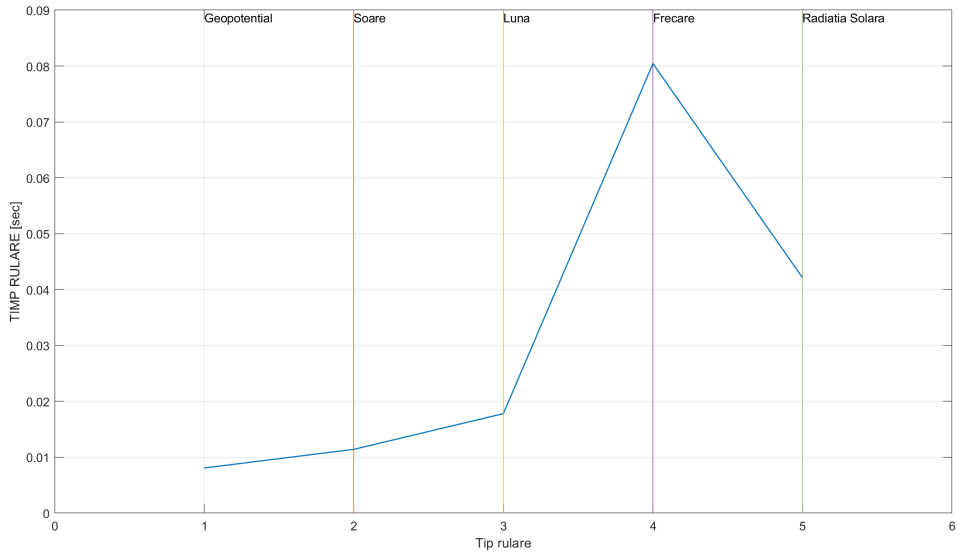


Figura 5: Eroarea mediilor timpilor de rulare dintre 2 iteraţii de simulări - pentru orbita GRACE-FO din 31 Decembrie 2021

În Tabelul 2 avem prezentate erorile relative ale timpilor de rulare.

Tip perturbaţie adaugată	Eroare relativă [%]
Geopotenţial	0.36
Atracţia Soarelui	0.48
Atracţia Lunii	1.04
Frecarea cu atmosfera	1.63
Radiaţia solară	0.98

Tabela 2: Eroarea relativă a mediilor timpilor de rulare dintre 2 iteraţii de simulări - pentru orbita GRACE-FO din 31 Decembrie 2021

Având în vedere datele prezentate anterior cel mai bun compromis dintre radicalul erorii medii pătratice şi viteza de propagare este reprezentat de setul de perturbaţii compus din: geopotenţialul Pământului, atracţia gravitaţională a Soarelui şi a Lunii.
În continuare, am realizat o analiză asupra radicalului erorii medii pătratice pentru propagarea orbitei satelitului GRACE-FO în diferite momente ale anului pentru a surprinde influenţa perturbaţiilor sezoniere asupra dinamicii acestui satelit. Influenţele sezoniere pot proveni de la distanţa variabilă faţă de Soare de-a lungul unui an ce influenţează atracţia gravitaţională exercitată de către acesta, radiaţia solară, dar şi frecarea cu atmosfera ce este dependentă de compoziţia atmosferei ce variază în funcţie de evenimentele meteorologice ce diferă de la un anotimp la altul.
Observăm în Figura 6 că timpii de rulare pentru diferitele scenarii de propagare a orbitei satelitului GRACE-FO au aceeaşi tendinţă, fapt ce ne demonstrează consistenţa simulărilor efectuate. În ceea ce priveşte radicalul erorii medii pătratice, observăm că în mare măsură tendinţa este de scădere odată cu creşterea numărului de perturbaţii. Pe baza acestor date putem valida afirmaţia anterioară conform căreia cel mai bun compromis între radicalul erorii medii pătratice şi viteza de propagare este

reprezentat de setul de perturbaţii compus din: geopotenţialul Pământului, atracţia gravitaţională a Soarelui şi a Lunii.

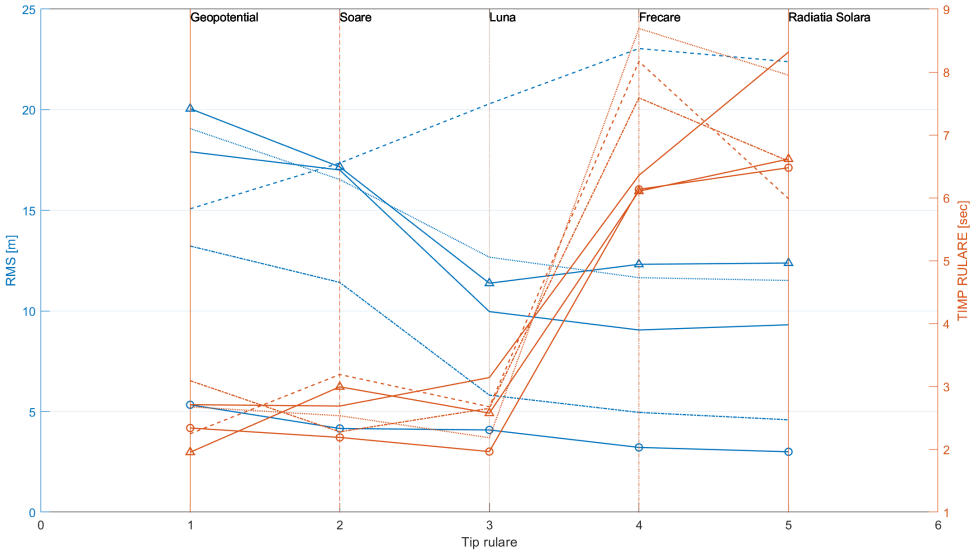


Figura 6: Radicalul erorii medii pătratice vs. timpul mediu de rulare pentru diferite configuraţii de perturbaţii - pentru orbite de-a lungul unui an

Configuraţia sistemului de calcul În tabelul următor vom prezenta configuraţia sistemului de calcul pe care am realizat etapa de simulare pentru ca cititorul să poată analiza rezultatele obţinute ţinând cont de puterea de calcul a sistemului utilizat.

Componentă sistem de calcul	Model utilizat
Procesor	AMD Ryzen 7 5800U 1.90 GHz
Memorie RAM	16 Gb
Memorie SSD	1 Tb
Placă grafică	Placă grafică Radeon integrată

Tabela 3: Configuraţia sistemului de calcul utilizat

2.2 Studiul algoritmilor de estimare

2.2.1 *Metode de estimare a traiectoriei*

Principalele metode utilizate în estimarea traiectoriei sunt următoarle:

1. Filtrul Kalman Liniar,
2. Filtrul Kalman Extins (Extended Kalman Filter - EKF),
3. Filtrul Kalman Inodor (Unscented Kalman Filter - UKF),
4. Filtrul Kalman Fara-Derivare (Derivative-Free Kalman Filter - DKF).

Filtrul Kalman Liniar Aplicarea Filtrului Kalman este asemănătoare aplicării metodei Recursive Least Squares. În timp ce Recursive Least Squares estimează unui parametru static, Filtrul Kalman are capacitatea de a estima o stare care evoluează în timp.

După cum și numele sugerează, filtrul Kalman Liniar se aplică sistemelor liniare

$$\begin{cases} \dot{x} = A_c x + B_c u, \\ z = C_c x. \end{cases} \tag{1}$$

Acestei dinamici îi putem atașa un estimator Kalman Liniar

$$\begin{aligned} \dot{\hat{x}} &= A_c \hat{x} + B_c u + K(\hat{y} - \gamma), \\ \hat{y} &= C_c \hat{x}, \end{aligned} \tag{2}$$

unde $\hat{x}$ - stările estimate ale sistemului, $\hat{y}$ - ieșirile estimate, iar K reprezintă amplificarea observatorului Kalman Liniar.

Pe scurt, Filtrul Kalman implementează atât un proces de Prediction Time Update, cât și un proces de Measurement Update. Etapa de Time Update implică utilizarea (în general, prin liniarizare) a ecuațiilor care dau dinamica sistemului, în timp ce etapa de Measurement Update presupune corectarea rezultatelor prin intermediul măsurătorilor.

Filtrul Kalman Extins Filtrul Kalman Extins este utilizat, în principal, pentru sistemele neliniare. Estimarea stării este efectuată în mod recursiv: modelul sistemului este liniarizat în mod continuu în jurul stării estimate. Ecuațiile care leagă dinamica sistemului de măsurători au forma

$$x(k+1) = \phi(x(k)) + L(k)u(k) + w(k), \tag{3a}$$

$$z(k) = \gamma(x(k)) + v(k), \tag{3b}$$

unde $w(k)$ și $v(k)$ sunt zgomotele specifice proceselor, ϕ - funcția care dă dinamica sistemului, γ - funcția care procesează măsurătorile corespunzătoare stării.

Filtrul Kalman Inodor Filtrul Kalman Inodor aparține unei clase mai largi de estimatoare neliniare, cunoscute ca Sigma-Point Kalman Filters.
Acesta folosește un procedeu denumit Unscented Transform (UT), care este construit în jurul principiului de a estima o probabilitate de distribuție în loc de o funcție neliniară.
Pentru etapa de Time Update, ecuațiile au forma

$$\begin{cases} [\hat{x}(k+1), P(k+1)^{\chi\chi}] = UT(f_d, \hat{x}(k), P(k)^{\chi\chi}), \\ P(k+1)^{\chi\chi} = P(k)^{\chi\chi} + Q(k), \end{cases} \tag{4}$$

în timp ce, pentru etapa de Measurement Update, ecuațiile sunt

$$\begin{cases} [\hat{z}(k), P(k)^{zz}, P(k)^{\chi z}] = UT(h_d, \hat{x}(k), P(k)^{\chi\chi}), \\ P(k)^{zz} = P(k)^{zz} + R(k). \end{cases}$$

Filtrul Kalman Derivative-Free Filtrul Kalman Derivative-Free poate fi aplicat pentru sisteme neliniare de forma

$$\dot{x} = f(x) + \sum_{i=1}^{rp} g_i(x)u_i + d_i \tag{5}$$

Utilizarea unui Filtru Kalman Derivative-Free implică transformarea sistemului într-o formă canonică Brunovsky, cu $r_1 + r_2 + \cdots + r_p = n$

$$\begin{cases} \dot{x}_1 & = x_2 \\ & \dots \\ \dot{x}_{r1-1} & = x_{r1} \\ \dot{x}_{r1} & = f_1(x) + \sum_{i=1}^{rp} g_{1j}(x)u_j + d_1 \\ \dot{x}_{r1+1} & = x_{r1+2} \\ \dots & \\ \dot{x}_{rp-1} & = x_{rp} \\ \dot{x}_{rp} & = f_p(x) + \sum_{i=1}^{p} g_{pj}(x)u_j + d_p \end{cases} \text{ şi } \begin{cases} y_1 = x_1 \\ \dots \\ y_p = x_{n-rp+1} \end{cases} \tag{6}$$

unde d_j este o variabilă asociată perturbațiilor externe.

Mai departe, pentru a obține ieșirea plată, transformăm sistemul prin aplicarea unei schimbări în spațiul stărilor

$$\zeta = T(x), T(x_0) = 0$$

și ajungem la sistemul

$$\begin{cases} \dot{\zeta} = A_c\zeta + B_c\nu(\zeta) \\ z = C_c\zeta \end{cases}$$

Acestei dinamici îi putem atașa un estimator Kalman Liniar

$$\begin{aligned} \dot{\hat{\zeta}} &= A_c\hat{\zeta} + B_c\nu(\zeta, t) + K(\hat{\gamma} - \gamma), \\ \hat{\gamma} &= C_c\hat{\zeta}, \end{aligned} \tag{7}$$

unde $\hat{\zeta}, \hat{\gamma}$ sunt stările și, respectiv, ieșirile estimate, în timp ce K este amplificarea observatorului Kalman Liniar.

2.2.2 *Criteriile de alegere a metodei de estimare potrivite*

Printre elementele care influențează calitatea estimării, se numără următoarele: pasul de eșantionare al simulării, valorile cu care este inițializat filtrul, rata de achiziție a datelor de la senzori, bias-ul măsurătorilor (în special la senzorii inerțiali), configurația solverului.

Configurația solverului trebuie să țină cont de evoluția dinamicii în timp, ceea ce înseamnă că atât metoda de integrare aleasă, cât și pasul de timp selectat trebuie să asigure faptul că nu există deviații puternice de la valoarea reală. Mai mult, rata de achiziție a datelor de la senzori trebuie sincronizată cu dinamica sistemului. În cazul în care dinamica este rapidă, iar preluarea măsurătorilor se face lent, putem avea erori semnificative.

Criteriile pentru selectarea metodei de estimare potrivite sunt următoarele:

1. gradul de neliniaritate al sistemului,
2. efortul de calcul al algoritmului, în special încărcarea procesorului,
3. acuratețea și precizia dorite,
4. rata de achiziție a datelor de la senzori,
5. spațiul disponibil pentru stocarea datelor.

Pentru utilizarea unei metode de liniarizare a sistemului, un factor important în alegerea metodei îl constituie cât de departe se află punctul de operare al sistemului de aproximarea liniară. În cazul în care dinamica sistemului este puternic neliniară, nu este recomandată o metodă bazată pe liniarizare, cum este EKF.

2.2.3 *Avantajele utilizării filtrului EKF*

Din cauza faptului că ecuațiile dinamicii satelitului sunt neliniare, aplicarea filtrului Kalman Liniar ar duce la rezultate eronate.

Datorită faptului că EKF aplică o liniarizare continuă a dinamicii în jurul stării estimate, stabilitatea numerică a acestuia se menține. Mai mult, efortul de calcul și, implicit, încărcarea procesorului sunt mai mici în cazul EKF decât pentru UKF.

2.3 Adaptarea algoritmilor pentru aplicaţii

2.3.1 *Ecuațiile Filtrului Kalman Extins*

Ne propunem să estimăm traiectoria unui satelit, reprezentată de starea $x(k)$, bazându-ne pe dinamica sistemului și pe disponibilitatea unor măsurători anterioare $y(k)$. Aplicarea unui filtru Kalman implică un proces iterativ care constă în două etape: predicție (propagare) și actualizare (corecție).
În cazul nostru, în care dinamica sistemului este puternic neliniară, estimarea stării este efectuată cu ajutorul Filtrului Kalman Extins (Extended Kalman Filter - EKF). Pentru aceasta, liniarizăm modelul sistemului în jurul stării înainte de a aplica procedeul de estimare.
Conform [10], în cazul unui sistem discret, algoritmul filtrului Kalman poate fi scris începând de la dinamica sistemului afectată de perturbații

$$x(k+1) = \phi(x(k)) + L(k)u(k) + w(k), \tag{8a}$$
$$z(k) = \gamma(x(k)) + v(k), \tag{8b}$$

unde $w(k)$ și $v(k)$ sunt zgomotele specifice proceselor, ϕ - funcția care dă dinamica sistemului, γ - funcția care procesează măsurătorile corespunzătoare stării.
Se consideră faptul că zgomotele de proces, $w(k)$ și $v(k)$, sunt albe, necorelate, au media zero și au matricile de covarianță cunoscute ($Q(k)$ și $R(k)$), respectiv

$$w(k) \sim (0, Q(k)), \quad v(k) \sim (0, R(k)), \tag{9}$$

şi

$$E[w(k) \cdot w(j)^\top] = Q(k) \cdot \delta(k-j), \tag{10}$$
$$E[v(k) \cdot v(j)^\top] = R(k) \cdot \delta(k-j), \tag{11}$$
$$E[v(k) \cdot w(j)^\top] = 0, \tag{12}$$

unde $\delta(k-j)$ este funcția delta Kronecker

$$\begin{cases} \delta(k-j) = 1, \text{dacă } k = j, \\ \delta(k-j) = 0, \text{dacă } k \neq j. \end{cases} \tag{13}$$

Trebuie menționat faptul că, deoarece EKF necesită liniarizarea modelului la fiecare pas de timp, acest lucru implică necesitatea calculului Jacobianului.
Astfel, este necesar ca ϕ și γ să fie suficient de netede încât să permită scrierea lor sub formă de serii Taylor.

Scriem $\phi(x)$ ca fiind vectorul de m funcții de tranziție a stării

$$\phi(x) = [\phi_1(x), \phi_2(x), \ldots . \phi_m(x)]^\top. \tag{14}$$

Similar, scriem $\gamma(x)$ ca fiind vectorul de p funcții

$$\gamma(x) = [\gamma_1(x), \gamma_2(x), \ldots, \gamma_p(x)]^\top. \tag{15}$$

Urmărind procedura de liniarizare, scriem ϕ și γ utilizând dezvoltarea în serii Taylor și aplicând Jacobianul funcțiilor $\phi(x)$ și $\gamma(x)$

$$\phi(x(k)) = \phi(\hat{x}(k)) + J_\phi(\hat{x}(k))[x(k) - \hat{x}(k)] + \dots, \tag{16a}$$
$$\gamma(x(k)) = \gamma(\hat{x}(k)) + J_\gamma(\hat{x}(k))[x(k) - \hat{x}(k)] + \dots \tag{16b}$$

Jacobianul lui $\phi(\hat{x}(k))$ se scrie ca

$$J_\phi(\hat{x}(k)) = \frac{\delta\phi}{\delta x}\bigg| x = \hat{x}^-(k) = \begin{pmatrix} \frac{\delta\phi_1}{\delta x_1} & \frac{\delta\phi_1}{\delta x_2} & \dots & \frac{\delta\phi_1}{\delta x_m} \\ \frac{\delta\phi_2}{\delta x_1} & \frac{\delta\phi_2}{\delta x_2} & \dots & \frac{\delta\phi_2}{\delta x_m} \\ \vdots & \vdots & \dots & \vdots \\ \frac{\delta\phi_m}{\delta x_1} & \frac{\delta\phi_m}{\delta x_2} & \dots & \frac{\delta\phi_m}{\delta x_m} \end{pmatrix}. \tag{17a}$$

Jacobianul lui $\gamma(\hat{x}(k))$ se scrie ca

$$J_\gamma(\hat{x}(k)) = \frac{\delta\gamma}{\delta x}\bigg| x = \hat{x}^-(k) = \begin{pmatrix} \frac{\delta\gamma_1}{\delta x_1} & \frac{\delta\gamma_1}{\delta x_2} & \dots & \frac{\delta\gamma_1}{\delta x_m} \\ \frac{\delta\gamma_2}{\delta x_1} & \frac{\delta\gamma_2}{\delta x_2} & \dots & \frac{\delta\gamma_2}{\delta x_m} \\ \vdots & \vdots & \dots & \vdots \\ \frac{\delta\gamma_m}{\delta x_1} & \frac{\delta\gamma_m}{\delta x_2} & \dots & \frac{\delta\gamma_m}{\delta x_m} \end{pmatrix}. \tag{18a}$$

2.3.2 *Implementarea EKF pentru estimarea traiectoriei*

Pornim de la următorul vector de stare

$$x(k) = [r_i(k), r_j(k), r_k(k), v_i(k), v_j(k), v_k(k)], \tag{19}$$

cu matricea de tranziție a stării

$$\phi(k) = [v_i(k), v_j(k), v_k(k), a_i(k), a_j(k), a_k(k)], \tag{20}$$

și matricea de observație

$$\gamma(k) = [r_i(k), r_j(k), r_k(k), v_i(k), v_j(k), v_k(k)]. \tag{21}$$

Pentru o estimare cât mai corectă a orbitei, matricea de tranziție a stării trebuie să ia în considerare toate forțele care acționează asupra satelitului.

Fiecare din forțele dezvoltate în sectiunea anterioară de modelare va avea un impact direct asupra estimării. În cazul în care am fi luat în calcul exclusiv influența efectului gravitațional al Pământului asupra mișcării satelitului, am fi obținut erori semnificative în procesul de estimare.

Astfel, elementele luate în considerare în dezvoltarea matricii de tranziție a stării sunt: forța de atracție dintre Pământ și satelit, geopotenţialul, accelerații gravitaționale datorate altor corpuri cereşti (în principal, Luna), presiunea exercitată de radiația solară (Solar Radiation Pressure - SRP) și efectul de încetinire dat de frecarea cu atmosfera.

$$\phi(x) = \phi_{Pamant-satelit}(x) + \phi_{geopotential}(x) + \phi_{Luna}(x) + \phi_{SRP}(x) + \phi_{atmosfera}(x). \tag{22}$$

Mai mult, e necesar să liniarizăm matricea de tranziție a stării pentru a putea efectua estimarea Kalman. Pentru acest lucru, vom calcula Jacobianul care conține derivatele parțiale ale ecuațiilor

mișcării pentru fiecare tip de perturbație

$$J_\phi = \begin{pmatrix} \frac{\delta v_i}{\delta r_i} & \frac{\delta v_i}{\delta r_j} & \frac{\delta v_i}{\delta r_k} & \frac{\delta v_i}{\delta v_i} & \frac{\delta v_i}{\delta v_j} & \frac{\delta v_i}{\delta v_k} \\ \frac{\delta v_j}{\delta r_i} & \frac{\delta v_j}{\delta r_j} & \frac{\delta v_j}{\delta r_k} & \frac{\delta v_j}{\delta v_i} & \frac{\delta v_j}{\delta v_j} & \frac{\delta v_j}{\delta v_k} \\ \frac{\delta v_k}{\delta r_i} & \frac{\delta v_k}{\delta r_j} & \frac{\delta v_k}{\delta r_k} & \frac{\delta v_k}{\delta v_i} & \frac{\delta v_k}{\delta v_j} & \frac{\delta v_k}{\delta v_k} \\ \frac{\delta a_i}{\delta r_i} & \frac{\delta a_i}{\delta r_j} & \frac{\delta a_i}{\delta r_k} & \frac{\delta a_i}{\delta v_i} & \frac{\delta a_i}{\delta v_j} & \frac{\delta a_i}{\delta v_k} \\ \frac{\delta a_j}{\delta r_i} & \frac{\delta a_j}{\delta r_j} & \frac{\delta a_j}{\delta r_k} & \frac{\delta a_j}{\delta v_i} & \frac{\delta a_j}{\delta v_j} & \frac{\delta a_j}{\delta v_k} \\ \frac{\delta a_k}{\delta r_i} & \frac{\delta a_k}{\delta r_j} & \frac{\delta a_k}{\delta r_k} & \frac{\delta a_k}{\delta v_i} & \frac{\delta a_k}{\delta v_j} & \frac{\delta a_k}{\delta v_k} \end{pmatrix}. \tag{23}$$

2.3.3 *Matricea de observație*

Calculul matricii de observație pentru măsurători Jacobianul matricii de observație are următoarea formă

$$\gamma = \begin{pmatrix} \frac{\delta r_i}{\delta r_i} & \frac{\delta r_i}{\delta r_j} & \frac{\delta r_i}{\delta r_k} & \frac{\delta r_i}{\delta v_i} & \frac{\delta r_i}{\delta v_j} & \frac{\delta r_i}{\delta v_k} \\ \frac{\delta r_j}{\delta r_i} & \frac{\delta r_j}{\delta r_j} & \frac{\delta r_j}{\delta r_k} & \frac{\delta r_j}{\delta v_i} & \frac{\delta r_j}{\delta v_j} & \frac{\delta r_j}{\delta v_k} \\ \frac{\delta r_k}{\delta r_i} & \frac{\delta r_k}{\delta r_j} & \frac{\delta r_k}{\delta r_k} & \frac{\delta r_k}{\delta v_i} & \frac{\delta r_k}{\delta v_j} & \frac{\delta r_k}{\delta v_k} \\ \frac{\delta v_i}{\delta r_i} & \frac{\delta v_i}{\delta r_j} & \frac{\delta v_i}{\delta r_k} & \frac{\delta v_i}{\delta v_i} & \frac{\delta v_i}{\delta v_j} & \frac{\delta v_i}{\delta v_k} \\ \frac{\delta v_j}{\delta r_i} & \frac{\delta v_j}{\delta r_j} & \frac{\delta v_j}{\delta r_k} & \frac{\delta v_j}{\delta v_i} & \frac{\delta v_j}{\delta v_j} & \frac{\delta v_j}{\delta v_k} \\ \frac{\delta v_k}{\delta r_i} & \frac{\delta v_k}{\delta r_j} & \frac{\delta v_k}{\delta r_k} & \frac{\delta v_k}{\delta v_i} & \frac{\delta v_k}{\delta v_j} & \frac{\delta v_k}{\delta v_k} \end{pmatrix} \tag{24}$$

Calitatea predicției este puternic influențată de factori precum: rata la care sunt achiziționate măsurătorile de la senzori, acuratețea măsurătorilor și numărul de măsurători disponibile.

2.3.4 *Procedeul iterativ de estimare Kalman*

Inițializarea filtrului Un element extrem de important în estimarea stării este inițializarea filtrului deoarece, în funcție de condițiile inițiale, este posibil să nu avem măsurători disponibile. Pentru estimarea inițială a stării înainte de a achiziționa măsurători utilizăm notația $\hat{x}(0)^+$. Astfel

$$\hat{x}(0)^+ = E(x(0)). \tag{25}$$

Pentru a calcula primele valori ale stării ($\hat{x}_1^-$), pornim de la condițiile inițiale și aplicăm (8) astfel

$$\hat{x}(1)^- = \phi(0)\hat{x}(0)^+ + L(0)u(0). \tag{26}$$

Calculul estimării bazate pe măsurători O modalitate de a calcula estimarea de stare **a priori** este să calculăm valoarea lui $x(k)$ bazată pe toate măsurătorile de dinainte de (dar nu inclusv) momentul de timp k. Dacă avem disponibile toate măsurătorile cu excepția celei de la momentul de timp k, putem calcula această estimare **a priori**, pe care o notăm $\hat{x}(k)^-$, respectiv

$$\hat{x}(k)^- = E[x(k)|y(1), y(2), ..., y(k-1)]. \tag{27}$$

După achiziționarea unei măsurători la momentul de timp k, ajungem la a avea disponibil un set de măsurători, adică $y(1), y(2), ..., y(k)$ și putem calcula o estimare **a posteriori**, notată ca $\hat{x}(k)^+$, respectiv

$$\hat{x}(k)^+ = E[x(k)|y(1), y(2), ..., y(k)]. \tag{28}$$

Estimările $\hat{x}(k)^-$ și $\hat{x}(k)^+$ sunt estimări ale aceleiași stări $x(k)$, însă $\hat{x}(k)^-$ este estimarea făcută înainte de efectuarea măsurătorii corespunzătoare momentului de timp k, în timp ce $\hat{x}(k)^+$ este estimarea făcută luând în calcul inclusiv măsurătorile de la pasul de timp k.

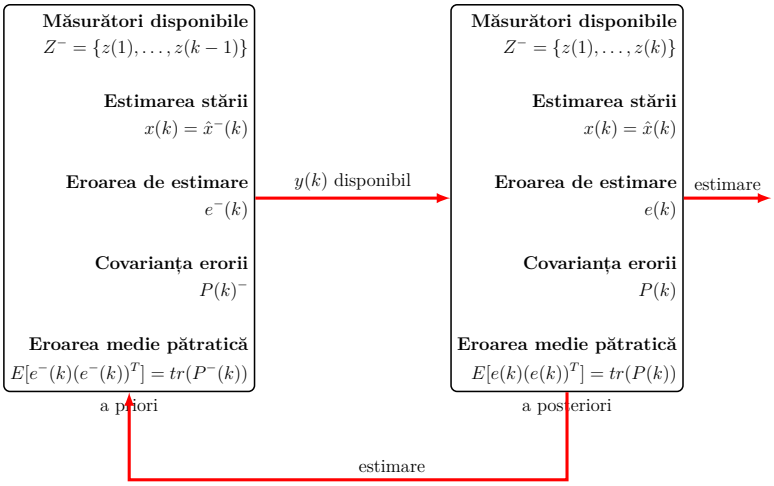


Figura 7: Diagrama EKF

În cazul în care ne dorim să găsim cea mai bună predicție a lui $x(k)$ la mai mult de un pas de timp înaintea efectuării măsurătorii, atunci putem calcula o **valoare predictată** a lui $x(k)$ astfel

$$\hat{x}(k|k-M)^- = E[x(k)|y(1), y(2), ..., y(k), ..., y(k-M)], \tag{29}$$

unde valoarea lui M depinde de informațiile disponibile sistemului nostru.
Alternativ, în cazul în care avem disponibile N măsurători după momentul de timp k, putem îmnunătăți **estimarea netezită** a stării la momentul k

$$\hat{x}(k|k+N)^- = E[x(k)|y(1), y(2), ..., y(k), ..., y(k+N)]. \tag{30}$$

Procesul iterativ al EKF Versiunea liniarizată a sistemului se scrie ca

$$x(k+1) = \phi(\hat{x}(k)) + J_\phi(\hat{x}(k))[x(k) - \hat{x}(k)] + w(k), \tag{31a}$$
$$z(k) = \gamma(\hat{x}^-(k)) + J_\gamma(\hat{x}^-(k))[x(k) - \hat{x}^-(k)] + v(k). \tag{31b}$$

De la momentul de timp $(k-1)^+$ până la momentul de timp k^- nu avem măsurători disponibile, iar, din acest motiv, estimăm starea pe baza informațiilor de dinamică de sistem reprezentate de matricea de tranziție a stării $\phi(k)$. Astfel, pentru etapa de predicție scriem

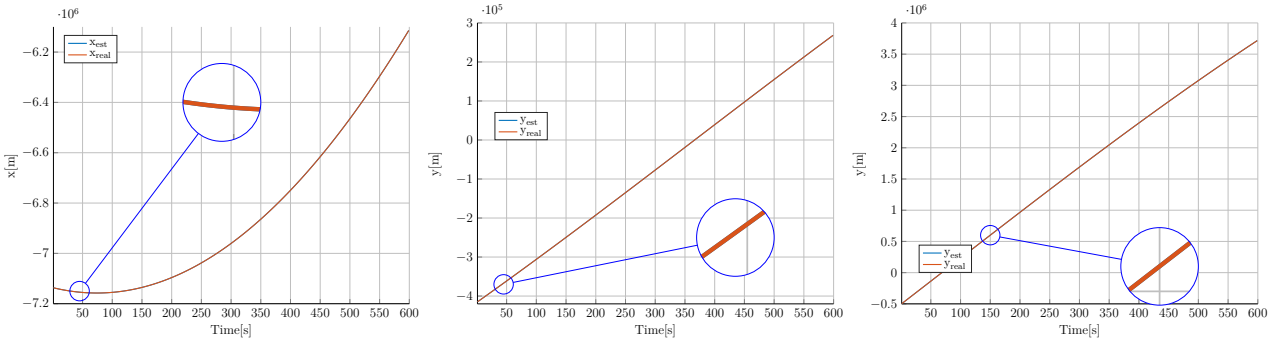
$$K(k) = P^-(k)J_\gamma(\hat{x}^-(k))[J_\gamma(\hat{x}^-(k))P^-(k)J_\gamma^\top(\hat{x}^-(k)) + R(k)]^{-1}, \tag{32a}$$
$$\hat{x}_k = \hat{x}^-(k) + K(k)[z(k) - \gamma(\hat{x}^-(k))], \tag{32b}$$
$$P(k) = P^-(k) - K(k)J_\gamma(\hat{x}^-(k))P^-(k). \tag{32c}$$

Pasul următor constă în actualizarea valorii lui P, covarianța erorii de estimare a stării. Astfel, lui $\hat{P}(0)^+$, covarianța estimării inițiale a lui $\hat{x}(0)$, îi putem atribui valoarea 0, în cazul în care cunoaștem exact starea inițială, sau $\infty \cdot I$, în cazul în care nu avem nicio informație legată de starea inițială. Pentru etapa de actualizare scriem

$$P^-(k+1) = J_\gamma(\hat{x}(k))P(k)J_\phi^\top(\hat{x}(k)) + Q(k), \tag{33a}$$
$$\hat{x}^-(k+1) = \phi(\hat{x}(k)) + L(k)u(k). \tag{33b}$$

2.3.5 *Rezultatele estimării*

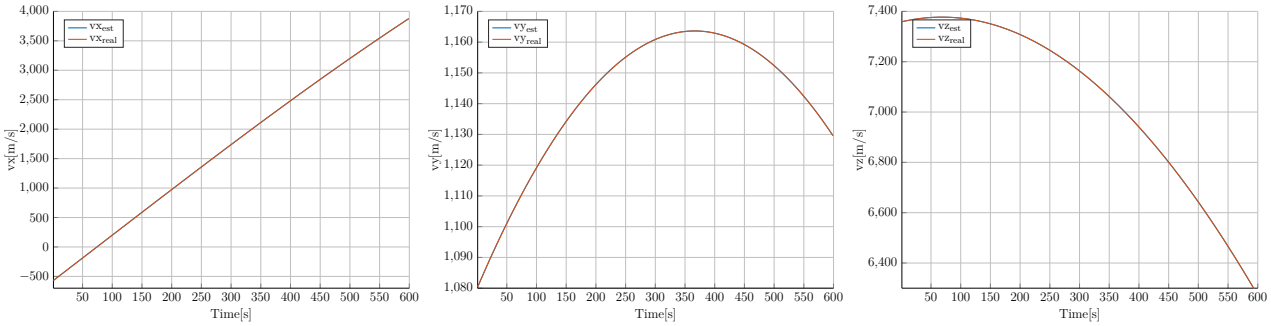
Rezultatele estimării pentru poziție Rezultatele estimării pentru poziție sunt reprezentate pentru fiecare axă



(a) Axa x (b) Axa y (c) Axa z

Figura 8: Rezultatele estimării pentru poziție

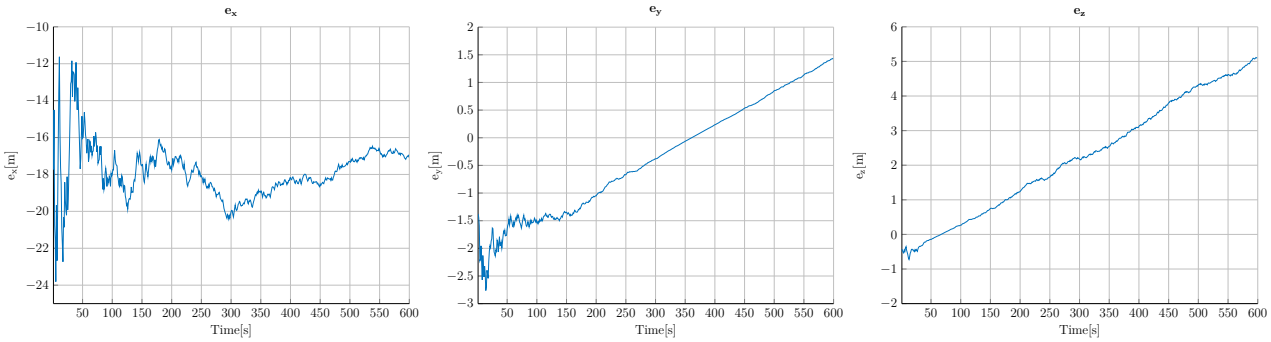
Rezultatele estimării pentru viteză Rezultatele estimării pentru viteză sunt reprezentate pentru fiecare axă



(a) Axa x (b) Axa y (c) Axa z

Figura 9: Rezultatele estimării pentru viteză

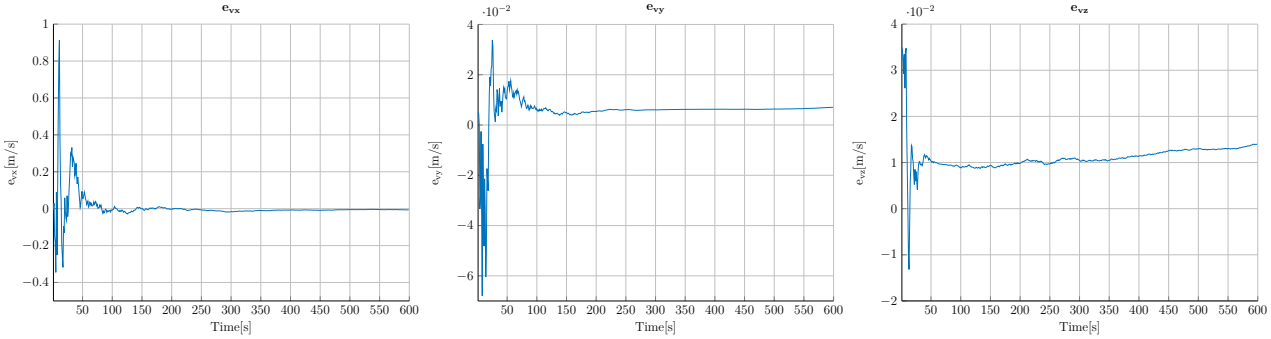
Erorile de estimare pentru poziție Erorile de estimare pentru poziție sunt reprezentate pentru fiecare axă



(a) Eroarea pe axa x (b) Eroarea pe axa y (c) Eroarea pe axa z

Figura 10: Erorile de estimare pentru poziție

Erorile de estimare pentru viteză Erorile de estimare pentru viteză sunt reprezentate pentru fiecare axă



(a) Eroarea pe axa x (b) Eroarea pe axa y (c) Eroarea pe axa z

Figura 11: Erorile de estimare pentru viteză

2.3.6 *Estimarea robustă a traiectoriilor*

Pentru un sistem neliniar dat în forma unei ecuații diferențiale standard

$$\dot{x}(t) = f(x(t), w(t)), \tag{34}$$

unde $x \in \mathbb{R}^n$ denotă starea internă a sistemului iar $w \in \mathbb{R}^m$ denotă perturbațiile ce afectează sistemul suntem interesați de caracterizarea comportamentului "worst-case". În continuare considerăm că perturbația este mărginită de o regiune convexă $W \subset \mathbb{R}^m$.

Pe baza presupunerii anterioare se pot construi aproximări interioare și exterioare ("inner" și "outer") ale regiunii ce acoperă traiectoriile posibile ale sistemului. Mai precis, putem construi mulțimea "forward reachable" definită pe intervalul $[t_i, t_f]$ astfel

$$\mathcal{R}(X_0, W, [t_i, t_f]) = \bigcup_{t \in [t_i, t_f]} \{x([t_i, t]; x_0, w(\cdot)) \in \mathbb{R}^n : \ w(\tau) \in W, \forall \tau \in [t_i, t], x(t_i) \in \mathcal{X}_0\}, \tag{35}$$

unde notația $x([t_i, t]; x_0, w(\cdot))$ înseamnă "traiectoria ce respectă dinamica (34), pornește din starea inițială $x(t_i)$ și este afectată de perturbația admisibilă $w(t)$".

În mod evident, calculul (35) este dificil. Implementarea uzuală este cea de calcul recursiv folosind aproximări repetate pentru a menține rezonabilă complexitatea reprezentării. Neliniaritatea lui (34) poate să afecteze puternic (în sens negativ) dificultatea de rezolvare a algoritmului. În continuare, vom folosi uneltele din CORA toolbox [9] ce, folosind descrieri zonotopice, sunt capabile să returneze reprezentări ale (35) de complexitate și într-un timp de calcul rezonabil.

Rolul construcției (35) este de a returna o mulțime variabilă în timp $\mathcal{R}$ care, în mod garantat, acoperă cele mai nefericite combinații de traiectorii ("worst-case analysis"). Grafic aceasta conduce la un "tub" ce este garantat să conțină toate traiectoriile, atâta timp cât modelul folosit este corect iar perturbația rămâne în limitele considerate a priori.

Chiar și folosind metode moderne [9], sunt necesare aproximări interioare și exterioare ale regiunii de interes pentru a putea ține sub control complexitatea reprezentării

$$\mathcal{R}^{\text{in}}(X_0, W, [t_i, t_f]) \subseteq \mathcal{R}(X_0, W, [t_i, t_f]) \subseteq \mathcal{R}^{\text{out}}(X_0, W, [t_i, t_f]), \tag{36}$$

unde $\mathcal{R}^{\text{in}}, \mathcal{R}^{\text{out}}$ denotă aproximările interioare și respectiv exterioare ale mulțimii $\mathcal{R}$ iar

$$\bar{x}([t_i, t]; x_0) := x([t_i, t]; x_0, 0) \in \mathcal{R}(X_0, W, [t_i, t_f]), \tag{37}$$

denotă traiectoria nominală (unde presupunem perturbație nulă pe intervalul curent: $w(t) = 0, \forall t \in [t_i, t_f]$). Din punct de vedere practic, cât timp diferența dintre aproximările interioară și exterioară

este rezonabilă (distanța dintre $\mathcal{R}^{\text{in}}$ și $\mathcal{R}^{\text{out}}$ nu depășește o limită impusă a priori), avem certitudinea unei aproximări corecte. Trebuie menționat că dimensiunea mulțimilor considerate este afectată nu doar de propagarea perturbațiilor cât și de erorile de aproximare ce apar din motive tehnice. Spre exemplu:

- dinamica neliniară a unui model este aproximată printr-un număr finit de termeni din seria sa Taylor (cu cât creștem numărul lor, cu atât crește precizia dar și complexitatea reprezentării),
- pentru dinamici continue, se consideră proceduri de discretizare unde, de-a lungul pasului de eșantionare, se presupune că perturbația rămâne constantă,
- pentru a menține constantă complexitatea reprezentării, se folosesc mulțimi de complexitate mai scăzută pentru mulțimea obținută în urma iterației curente ("order reduction methods").

În mod uzual traiectoriile rezultate sunt grupate în jurul traiectoriei nominale (37) și doar cazuri atipice (unde zgomotul și valoarea curentă a stării se suprapun în mod nefericit) conduc la traiectorii apropriate de marginea mulțimii (35).

2.3.7 *Modelarea traiectoriilor pentru satelit*

În continuare, considerăm cazul concret al dinamicii orbitale a satelitului

$$\ddot{r} = -\frac{\mu}{\|r\|_2^3} r + a_p\,, \quad r(0) = r_0\,, \dot{r}(0) = \dot{r}_0 \tag{38}$$

unde μ este constanta gravitațională iar $a_p = a_p(t)$ este perturbația ce afectează vectorul de accelerație. Relația (38) se poate aduce în forma standard (34)

$$\frac{d}{dt}\begin{bmatrix} r \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \dot{r} \\ -\dfrac{\mu}{\|r\|_2^3} r + a_p \end{bmatrix}, \tag{39}$$

ceea ce permite construcția mulțimii (35) utilizând [9].

Forma propusă în (39) este cea mai simplă conținând în model doar interacția satelit-planetă; termeni adiționali ce pot fi adăugați sunt, spre exemplu, cei ce corespund interacțiunii Pământ - Lună sau Pământ - Soare; spre exemplu, (39) devine

$$\frac{d}{dt}\begin{bmatrix} r \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \dot{r} \\ \mu\left(-\dfrac{r}{\|r\|_2^3} + \dfrac{d_{\text{sol}}}{\|d_{\text{sol}}\|_2^3} + \dfrac{h_{\text{sol}}}{\|h_{\text{sol}}\|_2^3}\right) + a_p' \end{bmatrix}, \tag{40}$$

unde d_{sol} și h_{sol} denotă distanța satelit – Soare și, respectiv, distanța Pământ – satelit. Obervăm că prin adăugarea acestui termen se reduce nivelul perturbației conform relației

$$a_p' = a_p - \mu\left(\frac{d_{\text{sol}}}{\|d_{\text{sol}}\|_2^3} + \frac{h_{\text{sol}}}{\|h_{\text{sol}}\|_2^3}\right). \tag{41}$$

În concluzie, modelul matematic ce descrie orbita se construiește în mod progresiv prin adăugarea unor noi termeni; dacă aceștia sunt considerați în model, acesta devine mai precis iar termeni ne-adăugați sunt considerați ca surse de eroare și contribuie la descrierea mulțimii admisibile.

Câteva observații sunt necesare în ceea ce privește termenii din perturbație:

- pentru fiecare dintre termenii perturbație este necesară mărginirea valorilor posibile ale acestora; rămâne un subiect deschis cum anume se selectează aceste margini:

  componentă deterministă spre exemplu, pentru termenii ce corespund influențelor "third body" mișcarea corpurilor cerești este cunoscută și predictibilă;

componentă stocastică spre exemplu, în LEO încă există contact cu atmosfera, iar frecarea cu aerul este, prin definiție, nedeterministă;

- în implementarea curentă se consideră o magnitudine a perturbației constantă; în realitate, aceasta este dependentă de timp; tehnic, această problemă se poate rezolva prin aproximarea variațiilor pe un interval de timp suficient de mic și rezolvarea iterativă a problemei.

Trebuie menționat că funcția folosită (REACH, a librăriei CORA) aproximează dinamica neliniară ((39) sau (40)) prin utilizarea primilor termeni ai seriei Taylor. Aceasta conduce rapid la formulări complexe ce cresc substanțial durata timpului de calcul. Cu alte cuvinte, se poate ajunge la un punct unde reducerea incertitudinilor nu mai este justificată de creșterea complexității de calcul.

Pentru clarificare, modul de lucru al funcției REACH este de a

- primi ca argument handle-ul funcției ce definește dinamica,
- urmată de transformarea acesteia într-o formă simbolică,
- pentru a putea calcula analitic derivatele (Jacobian, Hessian, și tensori de ordin superior).

Folosirea SYMBOLIC TOOLBOX, deși permite calcule simbolice, devine rapid nepractică (mai ales pentru folosirea unor funcții precum cele generate de biblioteca HIGH PRECISION ORBIT PROPAGATOR – HPOP). Ca urmare, studiem aplicarea CasADi [8], care execută în mod eficient proceduri de diferențiere automată. Cu alte cuvinte, este posibilă obținerea în mod mult mai eficient a derivatelor de ordin superior pentru o dinamică puternic neliniară. Prin urmare, suntem interesați de modificarea implementării existente astfel încât:

- modificăm funcția ce definește dinamica satelitului, așa cum reiese din HPOP, folosind variabilele simbolice SX introduse de CASADI;
- modificăm metoda REACH din CORA în așa fel încât să accepte apelările CASADI pentru construcția primilor termeni din seria Taylor asociată.

Demn de menționat este faptul că librăria CASADI are implementări echivalente în Python și C++. Cu alte cuvinte, putem folosi `MATLAB` pentru analiză și debugging iar pentru rapiditate și implementări pe platforme ce nu permit rularea `MATLAB`, putem folosi implementări în Python/C++.

2.3.8 *Ilustrații implementări practice*

Folosind parametrii cu valorile date în fragmentul de cod următor

```
Y0 = [-1837611.66201894;-3494142.56342754;-5622887.39840415;
3086.67801692727;5417.40886486767;-4369.10233250134];

params.tFinal = 40;
params.R0 = zonotope([Y0,eye(6)*1e2]);
params.U = zonotope([zeros(3,1),eye(3)*1e1]);

options.timeStep = 1;
options.taylorTerms = 4;
options.zonotopeOrder = 50;
options.intermediateOrder = 5;

options.errorOrder = 1;
options.alg = 'lin';
options.tensorOrder = 2;
```

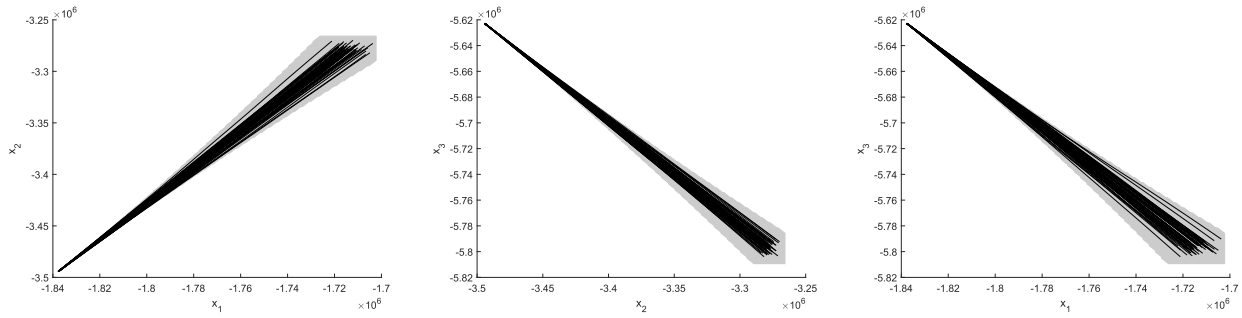
generăm (35) și un mănunchi de traiectorii obținute pentru diverse valori ale perturbației:

```
% System Dynamics
accel = nonlinearSys(@simplified_accel,6,3);

% Reachability Analysis
R = reach(accel, params, options);

% Simulation
params.tFinal = 40;

simOpt.points = 600;
simOpt.fracVert = 0.5;
simOpt.fracInpVert = 0.5;
simOpt.inpChanges = 6;
simRes = simulateRandom(accel, params, simOpt);
```

Obținem astfel, proiectate pe rând, pe câte două axe, ilustrațiile din Figura 12 unde observăm comportamentul dorit/așteptat: i) regiunea (35) este calculată și afișată (gri deschis) iar ii) traiectoriile (negru) se regăsesc în interiorul acesteia la toate momentele de timp.

Trebuie remarcate și aspectele negative ale acestei abordări:

- pentru un orizont de predicție de 40 secunde (cel ilustrat în figură), timpul de calcul este relativ mare: 5.04 secunde, ceea ce conduce la concluzia că nu putem implementa un astfel de mecanism la nivel de satelit (cel puțin, nu fără simplificări majore);
- funcționând în buclă deschisă, eroarea crește în mod exponential; iterând pe un orizont lung va conduce în consecință la valori ce sunt corecte dar extrem de conservative.



(a) proiecție $x - y$ (b) proiecție $y - z$ (c) proiecție $x - z$

Figura 12: Ilustrație mulțime admisibilă pentru intervalul $[0, 40]$

Pentru a ilustra mai clar mecanismul, în Fig 13 sunt evidențiate primele 5 componente (alternând culorile albastru și roșu) ale (35), proiectând pe subspațiul $x - y$.

Dincolo de problemele de implementare tehnică, dificultatea evidentă este creșterea dimensiunii tubului ce mărgineşte traiectoriile sistemului. Această evoluție este naturală și provine din integrarea în buclă deschisă a perturbației exogene a_p. Soluția este implementarea unui mecanism de corecție (idee uzuală de altfel în orice mecanism de estimație) pentru a reseta valoarea estimării la valoarea măsurată. Prin urmare propunem următoarea implementare:

1. alegem un orizont de integrare în buclă deschisă de lungime rezonabilă (spre exemplu $T = t_f - t_i = 50$ secunde); evident lungimea acestui interval poate fi schimbată sau chiar poate să fie variabilă în timp dacă observăm porțiuni ale traiectoriei unde erorile se acumulează mai rapid;
2. la terminarea unui interval de integrare, măsurăm poziția satelitului și resetăm estimația în mod corespunzător (în cazul nostru, folosind datele misiunii Grace); o alternativă este pur și simplu folosirea centrului regiunii ce bornează estimația la finalul intervalului de integrare;

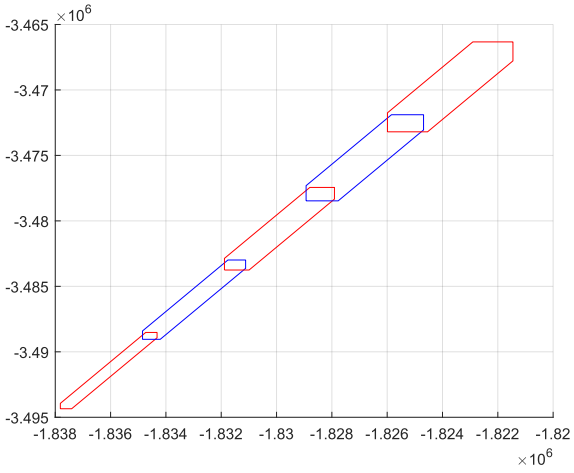


Figura 13: Detaliu din construcția mulțimii admisibile (proiecție pe sub-spațiul $x - y$, primele 5 componente)

3. repetăm procedura pentru pasul următor, până la terminarea orizontului de simulare.

Ideea acestui algoritm este schițată în fragmentul de cod `MATLAB` următor

```
grace = load("GNI1B_2022-01-ALL_C_04.mat");

interval=1 : 3600*2;
sample=1;

kmax = floor(interval(end)/params.tFinal);
for k=1:kmax
    R{k} = reach(accel, params, options);

    params.R0 = zonotope([grace.data(1+params.tFinal*sample*k,2:end)'...
                ,eye(6)*1e2]);
    simRes{k} = simulateRandom(accel, params, simOpt);
end
```

și ilustrată în Figura 14c pentru proiecția în planul $x - y$.

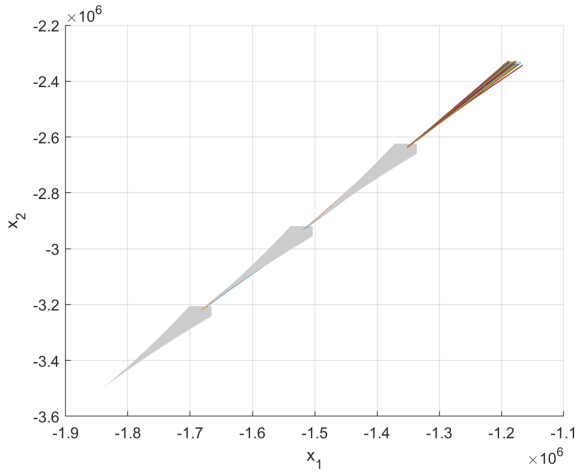
Figura 14a detaliază doar primele 3 iterații pentru a putea observa mai clar evoluția tubului ce acoperă traiectoriile posibile (se expandează exponențial pe măsură ce se avansează de-a lungul intervalului de integrare și este resetat atunci când se obține o nouă măsurătoare). Figura 14b ilustrează norma zonotopului obținut la capătul fiecărui interval de integrare. Acesta funcționează ca o măsură a erorii de integrare asociate unui interval. Observăm că, deși intervalele sunt de lungime egală, eroarea poate varia în timp.

Pentru această figură, parametrii folosiți și valorile obținute au fost:

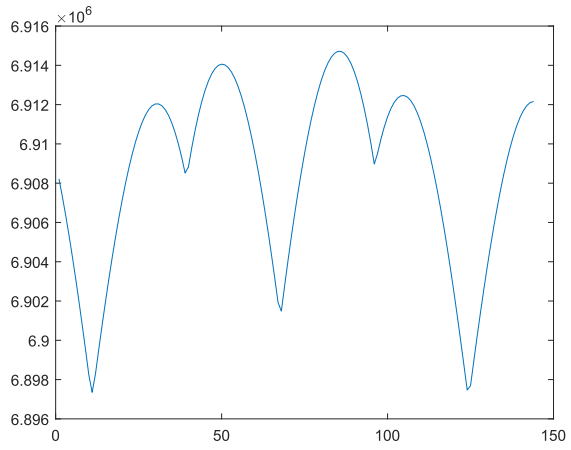
model	timp calcul	interval integrare	timp total	normă eroare tub (val. maximă)
eq. (39)	1032.32	50	7 200	6914708.98

Merită remarcat că procedura de integrare depinde de update-ul stării curente (i.e., măsurarea poziției și vitezei satelitului). Putem alege momentul la care solicităm o nouă măsurătoare (și implicit lungimea intervalului de integrare) ca un balans între costul unei noi măsurători (putere consumată de satelit pentru trimiterea și recepționarea unui mesaj) versus eroarea de aproximare ce crește exponențial cu lungimea pasului de integrare. Această analiză poate fi efectuată:

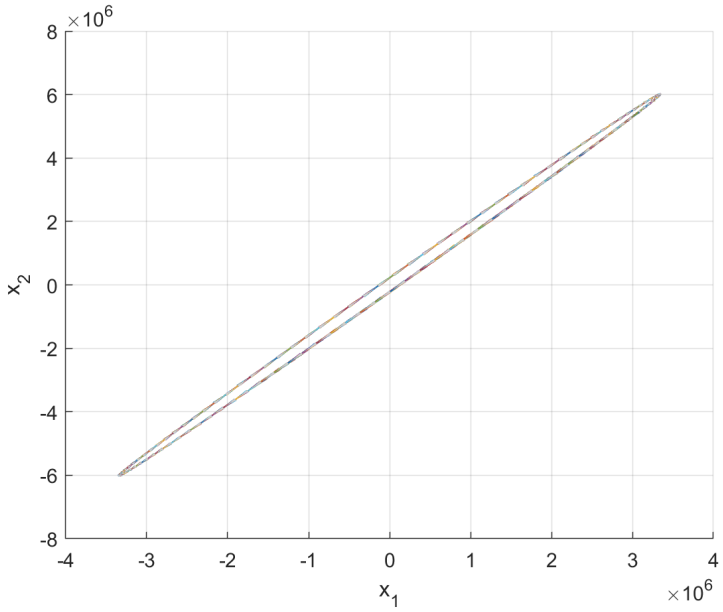
offline analiza se efectuează offline pe baza datelor deja existente (e.g., rulări anterioare din care putem extrage eroarea de integrare);



(a) detaliu integrare repetată



(b) normă tub la capătul intervalului de integrare



(c) integrare repetată

Figura 14: Ilustrație mulțime admisibilă cu integrare repetată și corecție, proiecție în planul $x - y$

adaptiv alegem o anumită valoare a erorii (i.e., diametrul tubului (35) depășește o valoare prescrisă, v. Figura 14b) pentru care să solicităm o nouă măsurătoare pentru corecția estimației.

2.4 Programarea algoritmilor în limbaje avansate

Pentru implementarea și integrarea algoritmilor dezvoltați este necesară conversia acestora din limbajele de nivel înalt (precum `MATLAB`) în limbaje de nivel scăzut (precum C/C++), pentru a putea fi portate pe dispozitivele hardware descrise în secțiunile de mai jos.

Prima tehnică abordată de grupul UB a fost conversia și portarea brută (”de mână”) a unui număr de funcții utile `MATLAB`, împreună cu verificarea dependențelor (printr-un instrument software de Dependency Analyzer) din întregul proiect.

Cu toate că această abordare poate funcționa în final, ea prezintă o serie de dezavantaje:

- Conversia codului necesită resurse de timp și efort semnificative.
- Procesul poate deveni foarte defectuos datorită procedurilor repetitive.
- După conversie, variantele ulterioare ale codului `MATLAB` necesită o nouă conversie.

- Întreaga echipă a proiectului PROFET preferă lucrul și experimentarea cu `MATLAB`.

Toate aceste aspecte au motivat utilizarea unui software cunoscut în comunitate: `MATLAB` Coder, care realizează conversia automată în C/C++. Fluxul de lucru este ilustrat în Figura 15.

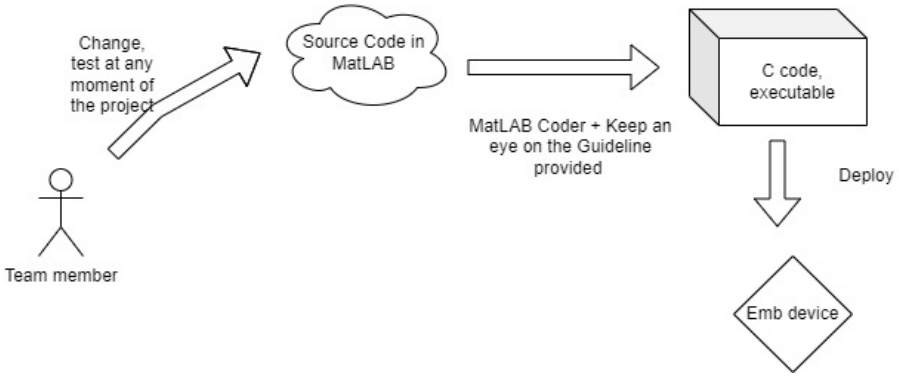


Figura 15

Avantajele concrete ale utilizării `MATLAB` Coder se rezumă pe scurt la: (*i*) eliminarea efortului repetitiv uman supus la erori; (*ii*) utilizarea unei interfețe Trace Code (vezi Figura 16) care asistă utilizatorul în evitarea problemelor care apar atunci când codul este foarte greu de citit. Cu toate aceste avantaje, codul `MATLAB` original necesită refactorizări semnificative pentru a putea fi compatibil cu portarea în C.

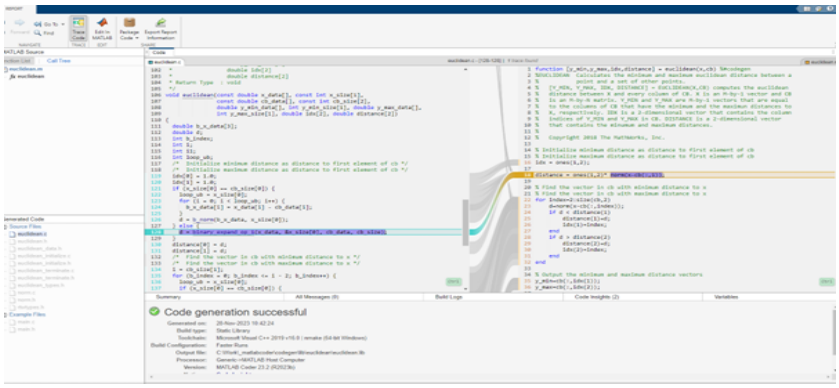


Figura 16

Ca exemplu, în Figura 17 s-a utilizat `MATLAB` Coder pentru conversia fișierului `Accel.m` în formatul original și s-au raportat erorile la conversie. Apariția erorilor nu semnalează un comportament defectuos al codului `MATLAB`, ci o formatare a codului care nu este compatibilă cu conversia directă în C.

De aceea, conversia întregului set de algoritmi din proiectul `h_pop` s-a realizat după refactorizarea codului `MATLAB` într-o formă supusă unui set de reguli care a permis evitarea erorilor precum cele exemplificate mai sus, i.e.:

- eliminarea tiparelor de matrici descrise explicit prin valori ale elementelor;
- eliminarea utilizării funcției `MATLAB` `feval` și înlocuirea acesteia cu apelul propriu-zis al funcției-argument;
- gruparea variabilelor de input/output în structuri specifice;
- tablourile necesită alocare dinamică ori statică *a priori* înainte de folosirea acestora;
- inițializarea tuturor variabilelor utilizate;

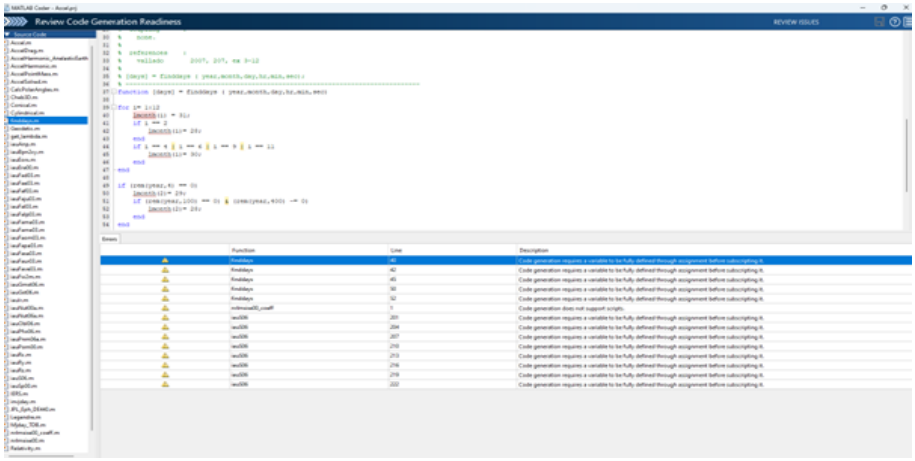


Figura 17

- separarea tuturor variabilelor și funcțiilor din script-uri;
- înlocuirea `str2num` cu `str2double`;
- înlocuirea funcției (de citire a fișierelor text); `fscanf` cu funcția asociată coder-ului pentru încărcarea fișierelor;

```
% read solar storm indices
%  ------------------------------------------------------------------------
% | YYYY DDD   JulianDay  F10   F81c  S10   S81c  M10   M81c  Y10   Y81c
%  ------------------------------------------------------------------------
% Replaced the load from file with loading with the coder component as needed for C compatibility. It is also much faster in matla
SOLdata = coder.load('SOLFSMY.txt', '-ascii');
SOLdata = transpose(SOLdata);
%fid = fopen('SOLFSMY.txt','r');
%SOLdata = fscanf(fid,'%d %d %f %f %f %f %f %f %f %f %f',[11 inf]);
%fclose(fid);
```

- dacă se utilizează tipul complex de date, portarea necesită specificarea explicită pentru a face diferența complex-real, de aceea fie: (i) se utilizează variable inițializate complex $(0, 0)$; sau (ii) se utilizează funcții de extragere a componentelor reale/imaginare pentru stocare;

```
for ii=1:6
    tline = fgetl(fid);

    tline_content = complex(0,0);
    tline_content = str2double(tline); % m and m/s
    Y0(ii) = real(tline_content);
end
tline = fgetl(fid);
AuxParam.area_solar = real(str2double(tline(49:end))); % m^2
tline = fgetl(fid);
```

- funcții `MATLAB` precum `rem` necesită, de asemenea, conversia argumentelor în numere reale;

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function a = iauFal03(t)

global const

% Mean anomaly of the Moon (IERS Conventions 2003).
tt = real(t);
a = rem(            485868.249036 +...
          tt * ( 1717915923.2178 +...
          tt * (         31.8792 +...
          tt * (          0.051635 +...
          tt * (        - 0.00024470 ) ) ) ), const.TURNAS ) * const.DAS2R;
```

- versiunea C (proprie `MATLAB` Coder) a factorizării LU necesită formatul matriceal dens, de aceea a fost necesară conversia din formatul rar în cel dens;

```
        B  = (valp(q2) + 1i*valp(q3))*Mass-Jac;

        B_full = full(B);
        [LL,UU,PP] = lu(B_full);  % NOTE: lu is supported only for dense matrices in C !
        L(:,:,q1)  = full(LL);
        U(:,:,q1)  = full(UU);
        P(:,:,q1)  = full(PP);
      end
    end
  else   % complex
```

- prealocarea zonelor buffer pentru citirea din fișier și eliminarea redundanțelor înainte de scrierea în buffer-ul final;

```
% Preallocating some of the arrays for compatibility with C and max
% performance
swdata = zeros(32, numrecsobs);
for i=1:numrecsobs
    linecontent = fscanf(fid,'%i %d %d %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %f %i %i %f %i %f
    swdata(:,i) = [linecontent(1:27);linecontent(29:33)];
end
```

Codul refactorizat `MATLAB` pentru compatibilitatea cu C este actualizat în branch-ul `hpop_mod` pe contul github al proiectului, mai precis la:

https://github.com/msstefan/profet-model-nl/tree/hpop_mod

De asemenea, întregul cod sursă C rezultat în urma conversiei totale a algoritmilor `MATLAB` se află la:

https://fmiunibuc-my.sharepoint.com/:u:/g/personal/ciprian_paduraru_fmi_unibuc_ro/EQ5rn952o35Is_6j49KteosBE73VQsO9P4epmUjGzzicOw?e=u43RZQ

Arhiva de la ultima adresă conține în directorul `HPOP/examples`, construit executabil folosind Cmake, atât pe Ubuntu cât și pe Windows, limbaj C, funcțional și pe *Raspberry Pi 4*.

Pentru compilarea-execuția proiectului C au fost luate în considerare următoarele aspecte:

- Uneori pachetul generat de `MATLAB` Coder nu conține un fișier extern. Acesta se poate obține în cazul nostru folosind utilitarul `packNGo` din `MATLAB`; vezi:

  https://uk.mathworks.com/help/coder/ref/packngo.html

- Apoi, pentru build-area pachetului a fost executat `HPOP_rtw.bat`, după care s-a facut proiect de Cmake în environment-ul de Clion.

- Pașii de mai sus nu sunt necesari dacă se foloseşte același proiect-template din arhivă.

2.5 Definirea specificaţiilor hardware

Începem analiza misiunii unui nanosatelit propunând trei tipuri de orbite joase în jurul Pământului, pentru a simula puterea generată de panourile solare din dotarea Cubesat-ului 6U. Orbitele sunt descrise în continuare şi sunt ilustrate în Figura 18.

GRACE_FO_Orbit

- Semi-axa majoră: 6886.1375 km
- Excentricitate: 0 (orbită circulară)
- Inclinație: 89°
- Atitudine: axa Z (ECI) aliniată cu Soarele

ISS_Orbit

- Semi-axa majoră: 6838.1365 km
- Excentricitate: 0 (orbită circulară)
- Inclinație: 51.6°
- Atitudine: axa Z (ECI) aliniată cu Soarele

SSO

- Semi-axa majoră: 7078.1375 km
- Excentricitate: 0 (orbită circulară)
- Inclinație: 97°
- Atitudine: axa Z (ECI) aliniată cu Soarele

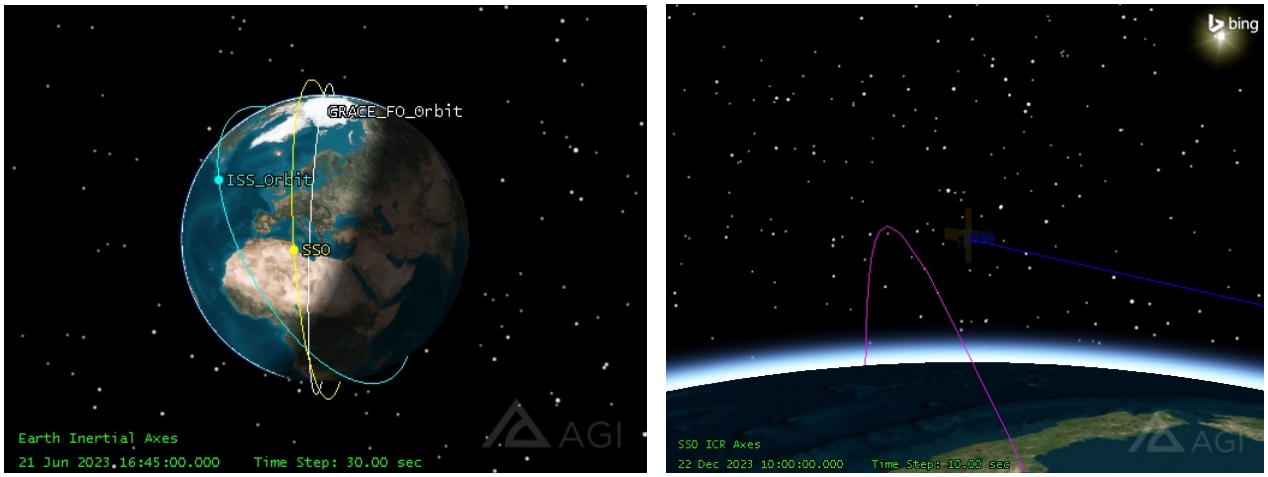


Figura 18: Cele trei orbite analizate: GRACE_FO_Orbit (cu alb), ISS_Orbit (cu cyan), şi SSO (cu galben). În imaginea din dreapta se observă atitudinea satelitului, care este orientat cu panourile către Soare.

- Ora locală a nodului descendent: ”06h:00.00” (orbită sincronă cu Soarele)

În cadrul analizei efectuate pentru orbita GRACE_FO_Orbit, observăm că valorile medii ale puterii generate în timpul solstițiului de iarnă, echinocțiului de primăvară și solstițiului de vară sunt respectiv $58.77W$, $40.89W$ şi $54.83W$. Calculând media acestor trei valori, obținem o putere medie aproximativă de $51.50W$, cu o deviaţie standard de $9.39W$.

Pentru orbita ISS_Orbit, obţinem rezultate puţin mai slabe faţă de cazul anterior, şi anume $41.14W$, $40.48W$ şi $40.46W$, respectiv pentru solstițiul de iarnă, echinocțiul de primăvară și solstițiul de vară. Media acestor valori este de aproximativ $40.69W$, cu o deviaţie standard de $0.39W$.

Nu în ultimul rând, orbita SSO ne oferă cele mai bune performanţe. Orbita sincronă cu Soarele este ideală pentru maximizarea energiei generate de panourile solare, deoarece asigură o expunere constantă și eficientă la lumina solară. Sateliții pe această orbită beneficiază de o iluminare solară relativ constantă pe tot parcursul zilei, menținând un unghi optim față de Soare pentru o captare eficientă a luminii. De asemenea, această orbită reduce semnificativ perioadele de umbrire cauzate de Pământ, oferind o expunere solară mai lungă și mai consistentă, ceea ce este esențial pentru sateliții care depind de energie solară stabilă și continuă. Puterile estimate sunt $65.86W$, $64.91W$ şi $51.33W$, cu o deviaţie standard de $8.13W$.

Rezultatele celor trei simulări sunt ilustrate în Figura 19.

Pe baza analizei efectuate se constată că puterea generată de panourile solare pe diferite orbite este suficientă pentru a susține funcționarea unui microcontroller mai performant. În particular, orbitele GRACE_FO_Orbit și ISS_Orbit prezintă o putere medie generată care depășește pragul minim necesar pentru operarea eficientă a microcontrollerelor utilizate în aplicații spațiale. Mai impresionant este faptul că orbita SSO depășește aceste valori, evidențiind un potențial semnificativ pentru integrarea unor microcontrollere avansate, ce necesită o putere mai mare. Această capacitate sporită de generare a energiei solare deschide noi posibilități, permițând utilizarea de sisteme de procesare mai sofisticate care pot îmbunătăți semnificativ colectarea și procesarea datelor în timp real.

În Tabelul 4 sunt prezentate specificațiile procesoarelor din seria i.MX 6 de la NXP. Aceste procesoare oferă o gamă variată de niveluri de performanță, cu opțiuni de single-core, dual-core și quad-core bazate pe arhitectura Arm Cortex. Sunt potrivite pentru o serie largă de aplicații datorită numeroaselor caracteristici, precum diverse interfețe de comunicare, management integrat al energiei și securitate avansată. Tabelul surprinde caracteristicile cheie standard.

Motivele principale pentru trecerea la microcontrollerul NXP i.MX6 au fost necesitatea îmbunătățirii performanței sistemului, alături de flexibilitatea și potențialul de extindere al sistemului final. Microcontrollerele clasice nu au putut îndeplini cerințele stricte de timp stabilite pentru algoritmi. În plus, decizia de migrare a fost consolidată de existența unor framework-uri și biblioteci consacrate (ex., bliblioteci precum LAPACK sau BLAS).

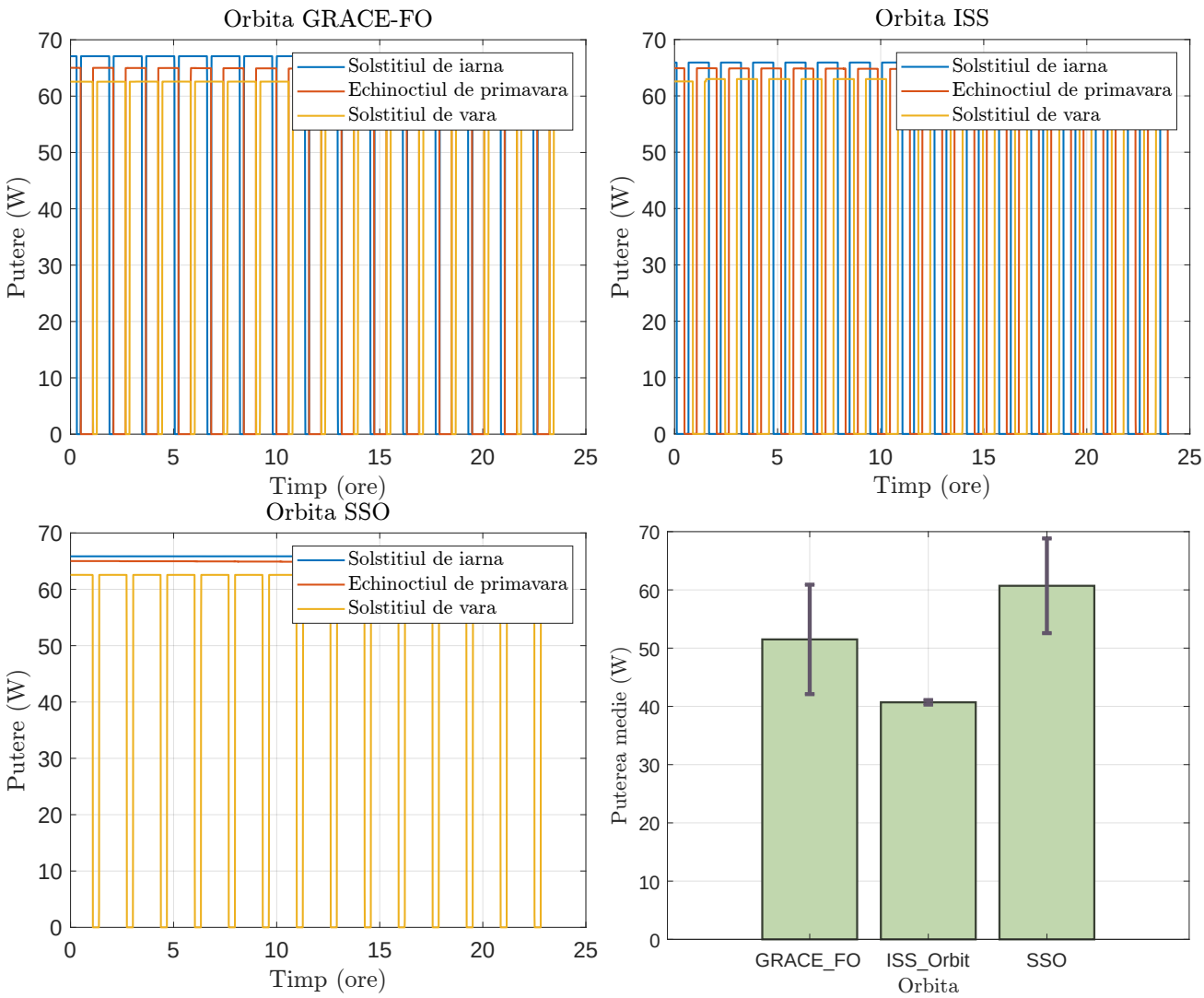


Figura 19: Analiza comparativă a puterii generate de panouri solare pe trei orbite diferite.

Caracteristică	**Descriere**
CPU	Multicore Arm Cortex-A7/A9/M4
Interfaţă de memorie	DDR
Protocol de comunicare	UART, SPI, I²C, I²S, CAN
Timer	Da
RTC securizat	Da
Managmenet de putere	Integrat
Interfeţe de afişare	Da
Ethernet	Da
Controller USB	Da
Benzi PCIe	Da
MMC/SDIO	Da
Canale ADC	Disponibil

Tabela 4: Specificaţii hardware cheie ale procesoarelor din seria NXP i.MX 6

2.6 Proiectarea circuitului electronic

Placa de procesare NXP i.MX6 se interfațează cu placa principală OrbFix prin intermediul conectorului de stivuire PC104 și comunică cu aceasta folosind protocolul UART. Conectorul, cât şi placa de procesare, sunt ilustrate în Figura 20. De asemenea, conexiunile electrice de pe circuitul electronic

sunt ilustrate în Figura 21.

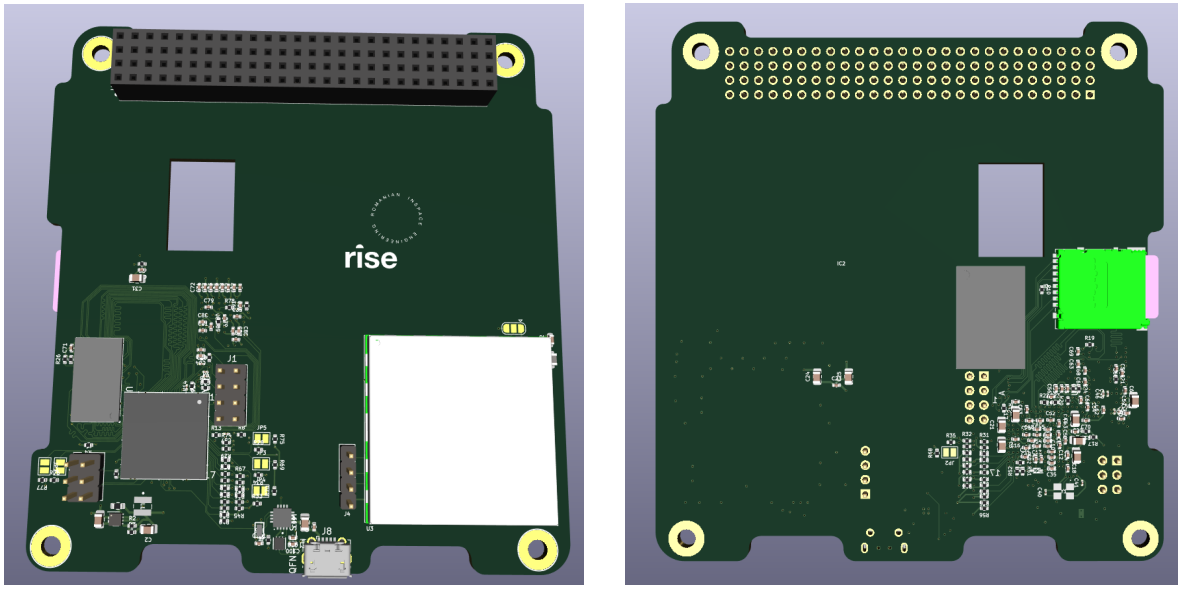


Figura 20: Schema 3D la scară a circuitului electronic.

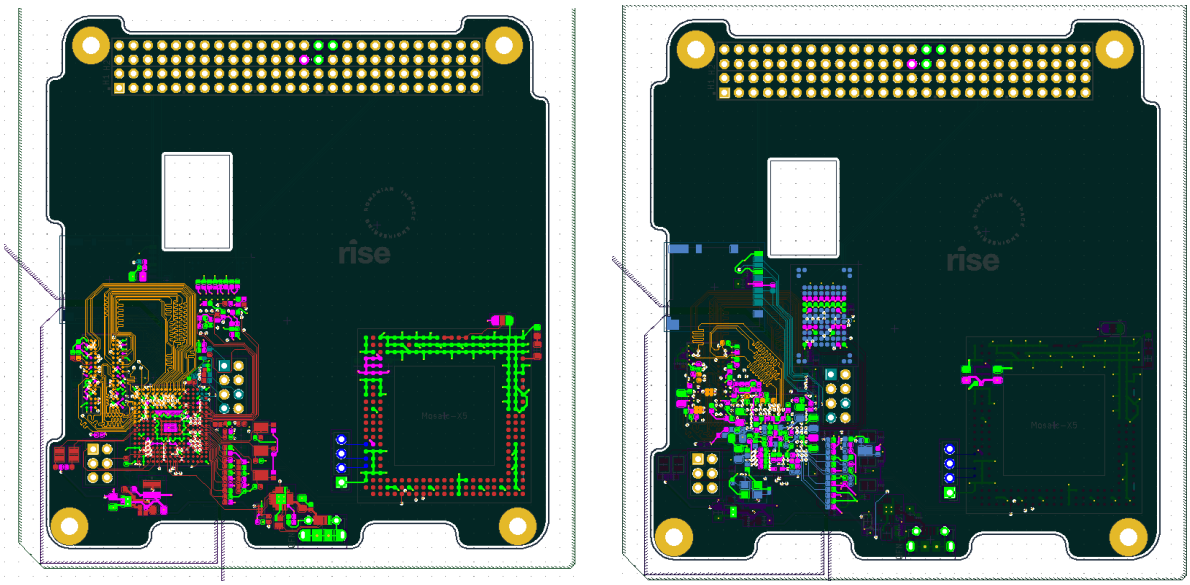


Figura 21: Circuitul electronic.

Arhitectura hardware modificată, adică trecerea de la o arhitectură SPARC-V8 la un sistem mai complex precum microcontrollerul NXP i.MX6 care rulează sistemul de operare Yocto, implică îmbunătățiri majore în ceea ce privește modularitatea și scalabilitatea. Această tranziție permite implementarea unei arhitecturi modulare, unde modulele de procesare pot fi dezvoltate, testate și înlocuite mai ușor datorită straturilor de abstractizare oferite de sistemul de operare. Mai mult, utilizarea setului extins de instrumente Yocto permite personalizarea și scalarea algoritmilor de procesare, deoarece poate suporta o gamă variată de componente software și biblioteci optimizate pentru capabilitățile hardware ale i.MX6.

În plus, acest design aduce flexibilitate în depanarea funcțională. Datorită faptului că sistemul de operare al microcontrollerului i.MX6 suportă SSH, se poate efectua depanarea la distanță cu ușurință, folosind un server SSH pentru a interacționa cu fișierele și folderele de pe sistemul îndepărtat, ceea ce face procesul de "debugging" mai eficient și mai puțin consumator de timp.

Principalele componente ale plăcii electronice sunt: un procesor NXP i.MX 6ULL cu un nucleu Arm Cortex®-A7 de 900 MHz, memorie SDRAM – DDR3L de 4GB, memorie flash NAND de 32Gbit și un

modul redundant de recepție GNSS Mosaic-x5 de la Septentrio. Placa include, de asemenea, un slot pentru card SD.

Placa de procesare comunică cu placa principală folosind UART (Universal Asynchronous Receiver/Transmitter) printr-un protocol de comunicaţie proprietar, care este o structură personalizată pentru organizarea și transmiterea datelor. Acest cadru proprietar este adaptat pentru a asigura un schimb eficient de date între plăci. Include un ID pentru antet, un ID pentru subsistemul plăcii de procesare și verificarea CRC. Utilizarea acestui cadru de date asigură buna funcţionare a sistemului, oferind o metodă sigură și fiabilă de transfer a informațiilor. În mod uzual, placa de procesare rămâne în stare latentă și se activează doar la primirea unei instrucțiuni specifice. Odată ce o comandă este emisă, sistemul execută acțiunea necesară și, ulterior, comunică rezultatele înapoi sistemului principal folosind protocolul UART.

Următoarele pagini includ schema electrică a microcontrollerului, detaliată pentru componentele principale şi pentru periferice.

Specificaţie	**Descriere**
Producător	NXP
Nucleu (1x)	ARM Cortex A7
Lățimea de date a magistralei	32 bit
Frecvență maximă de ceas	792 MHz
Buffer pentru Instrucțiuni L1 C	32 kB
Buffer pentru Date L1 C	32 kB
Tensiune de alimentare	Între 1.375 V și 1.5 V
Domeniu termic de operare	Între - 40 C și +105 C
Dimensiune memorie RAM	128 kB
Dimensiune memorie ROM	96 kB
Tip de instrucțiuni	Virgulă mobilă (floating point)
Interfețe	CAN, ESAI, Ethernet, I2C, SAI, SPI, UART, USB
Memorie tampon L2iInstrucțiuni / date	128 kB
Tip de memorie	DDR3 SDRAM, DDR3L SDRAM, LPDDR2 DRAM
Număr contoare de timp	4
Serie procesor	i.MX 6ULL

Tabela 5: Caracteristicile procesorului MCIMX6Y2CVM08AB.

2.7 Integrarea hardware/software a sistemului

Pentru minimizarea riscurilor în implementare, dezvoltarea software se face în trei etape succesive:

1. dezvoltarea și evaluarea performanțelor în sisteme de calcul de tip PC,
2. implementarea la nivelul unui hardware reprezentativ,
3. implementarea la nivelul hardware-ului echivalent de zbor.

Integrarea hardware și software a sistemului corespunde ultimelor două dintre aceste etape.

Sistemul dezvoltat în cadrul proiectului se bazează pe un dispozitiv de poziționare pentru sateliți dezvoltat de RISE care integrează un receptor GNSS și un microcontroller de tip LEON3, rezistent la radiații. Acesta din urmă nu poate executa în timp real algoritmii de predicţie a poziţiei satelitului, dezvoltați în cadrul proiectului PROFET. Este astfel necesară augmentarea puterii de calcul cu un modul de procesare.

Inițial s-a verificat transpunerea unor astfel de algoritmi utilizând un microcontroller din familia STM32. Acesta operează cu performanțe superioare, dar nu este la fel de flexibil și nu permite scalarea pe viitor la sisteme care să integreze mai multe tipuri de date și senzori. Astfel, pentru alinierea la evoluția rapidă din domeniul sateliților mici pentru dezvoltarea unui sistem care să rămână relevant, s-au căutat alternative scalabile pe termen mediu.

Ulterior am constatat că evoluția performanțelor sistemelor de procesare, manifestată în principal prin creșterea puterii de calcul și reducerea consumului de energie, permite includerea de procesoare de generație nouă (arhitectură ARM Cortex-A7) pe subsisteme destinate nanosateliților.

Caracteristicile procesorului utilizat (MCIMX6Y2CVM08AB) sunt listate în Tabelul 5.

Din punctul de vedere al alimentării cu tensiune, procesorul permite utilizarea mai multor moduri de funcționare de consum redus în care puterea consumată este de ordinul zecilor de mW, putând coborî sub 1 mW.

Pentru a facilita dezvoltarea simultană de software de către partenerii implicați în proiect, s-au achiziționat plăci de dezvoltare NXP MCIMX6ULL-EVK (Figura 29) care să permită implementarea iterativă a codului pe procesor, înainte de integrarea pe hardware-ul specific de zbor. Aceasta are avantajul că prezintă dezvoltatorului o serie de periferice și funcționalități adiționale ce grăbesc procesul de implementare și depanare. Astfel sistemul rulează o versiune de Linux, permițând instalarea facilă a pachetelor, bibliotecilor și altor dependințe necesare pentru compilarea și executarea codului.

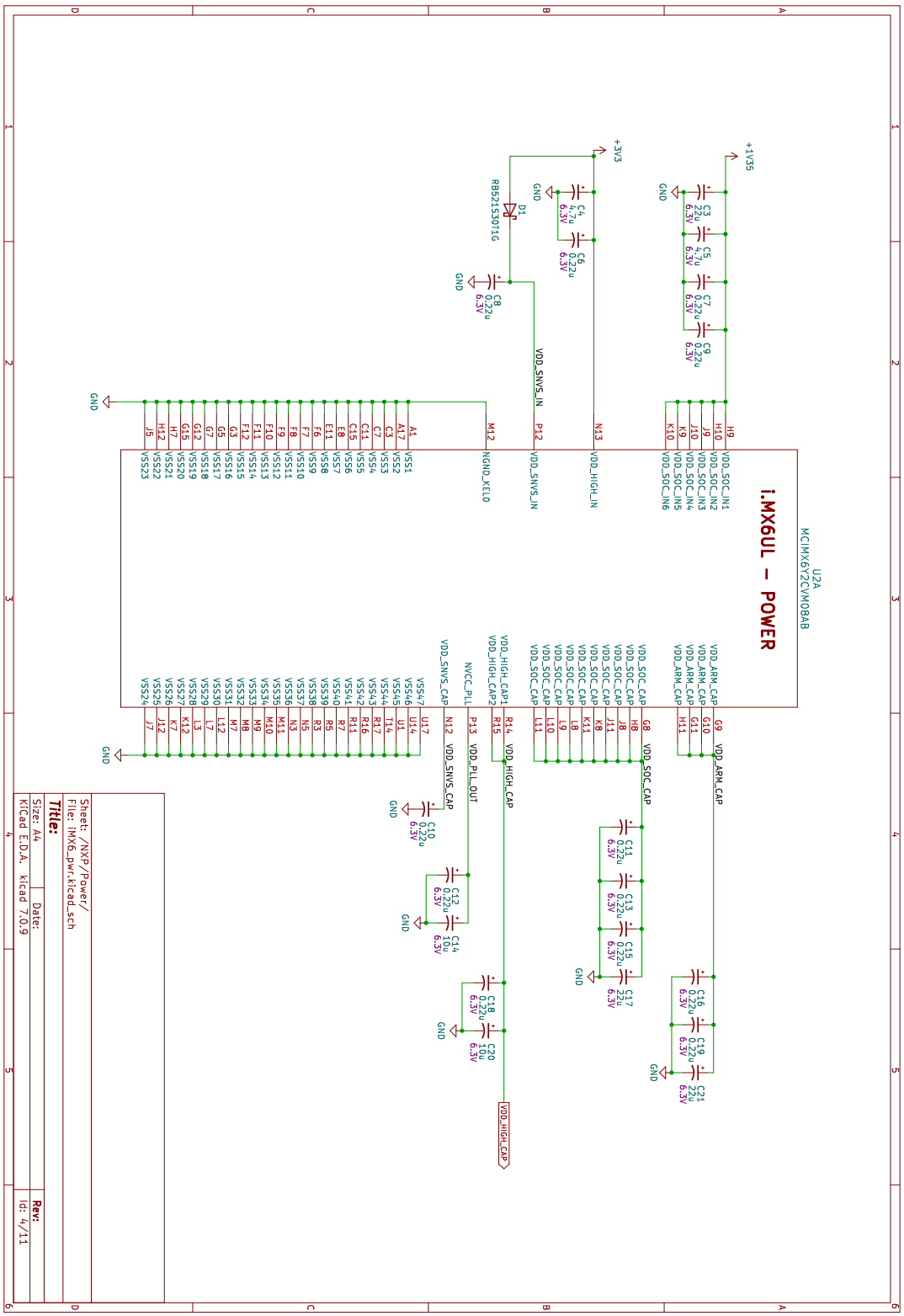


Figura 22: Schema electrică a NXP i.MX 6ULL – Alimentare

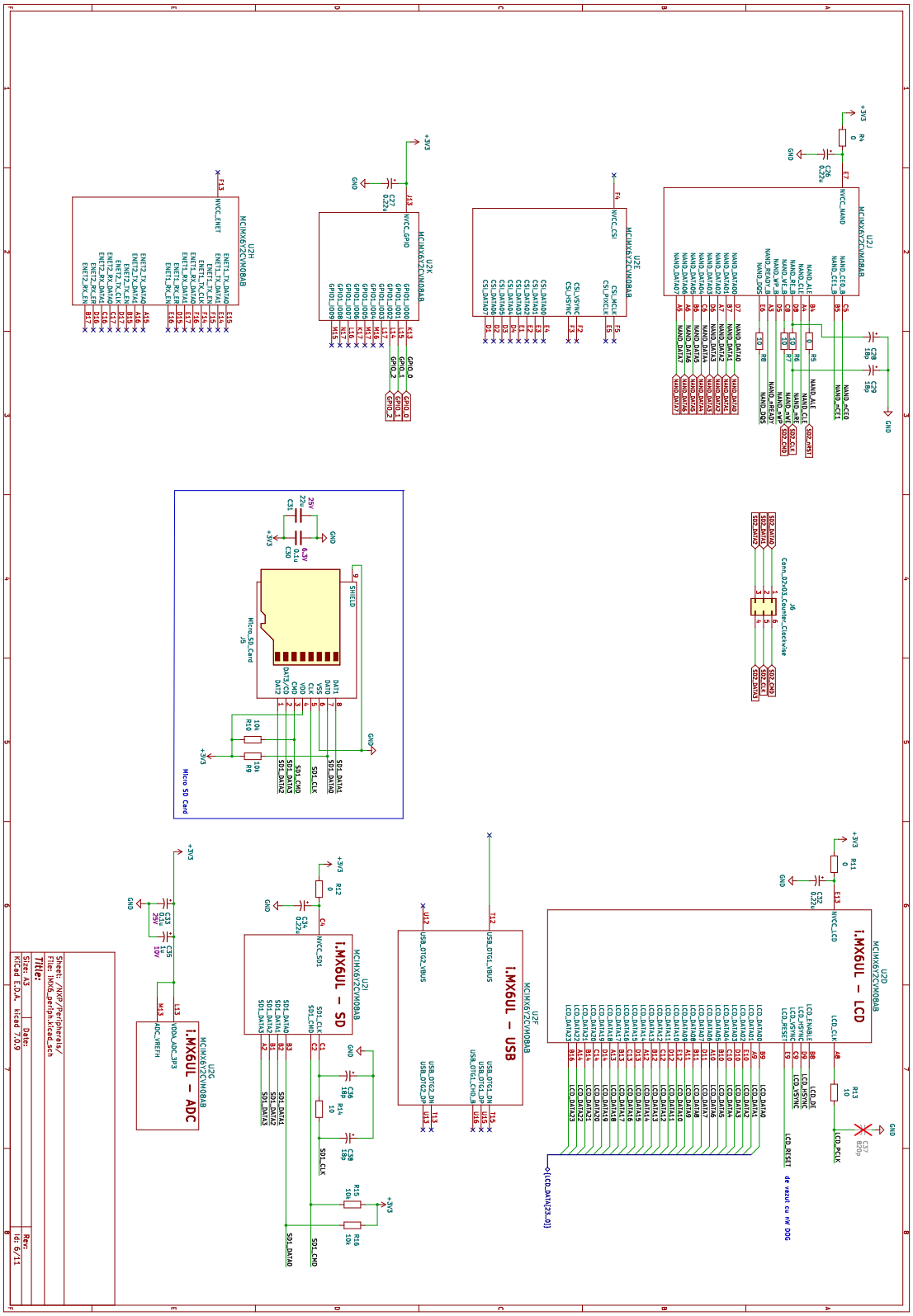


Figura 23: Schema electrică a NXP i.MX 6ULL – Periferice

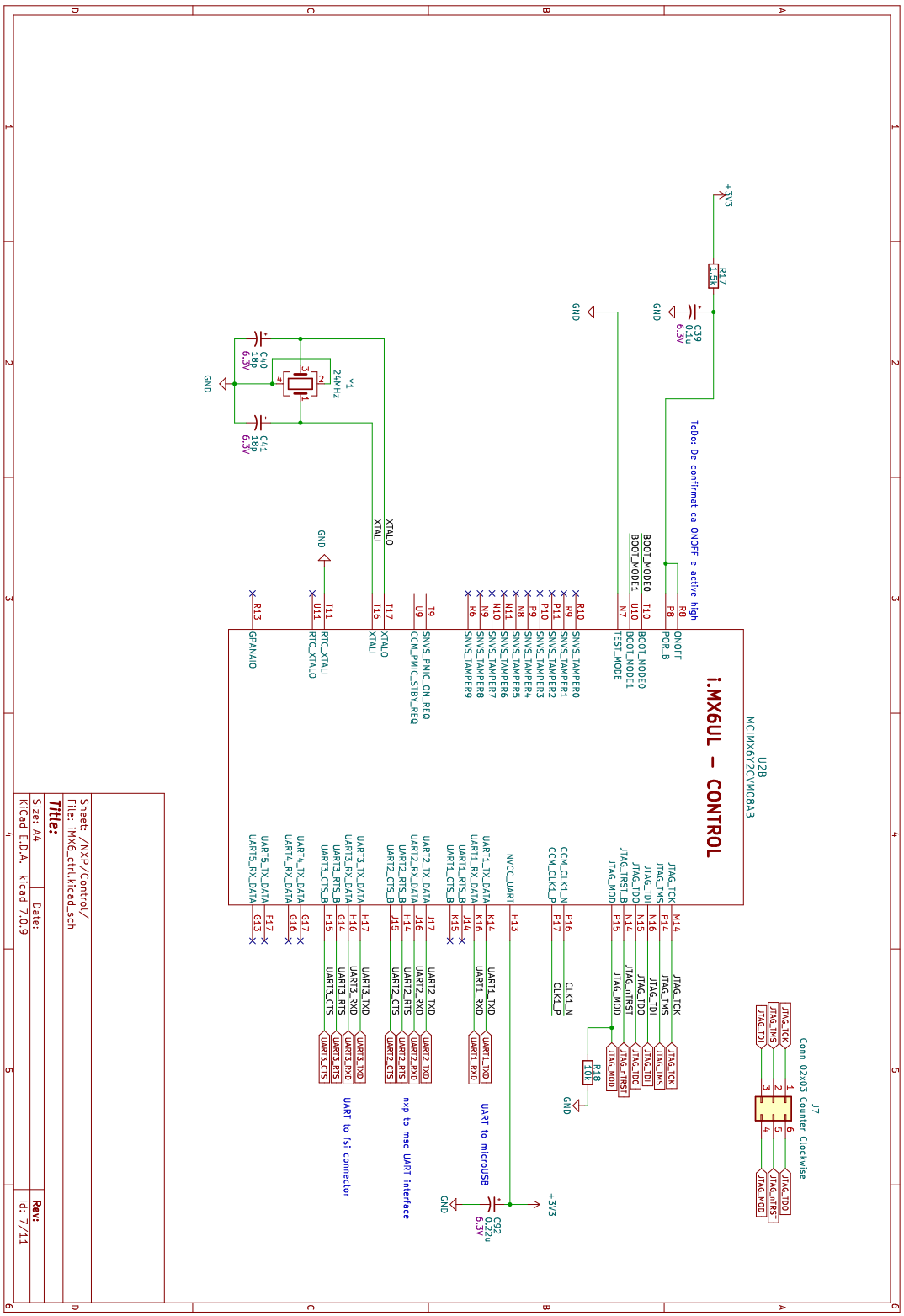


Figura 24: Schema electrică a NXP i.MX 6ULL – Control

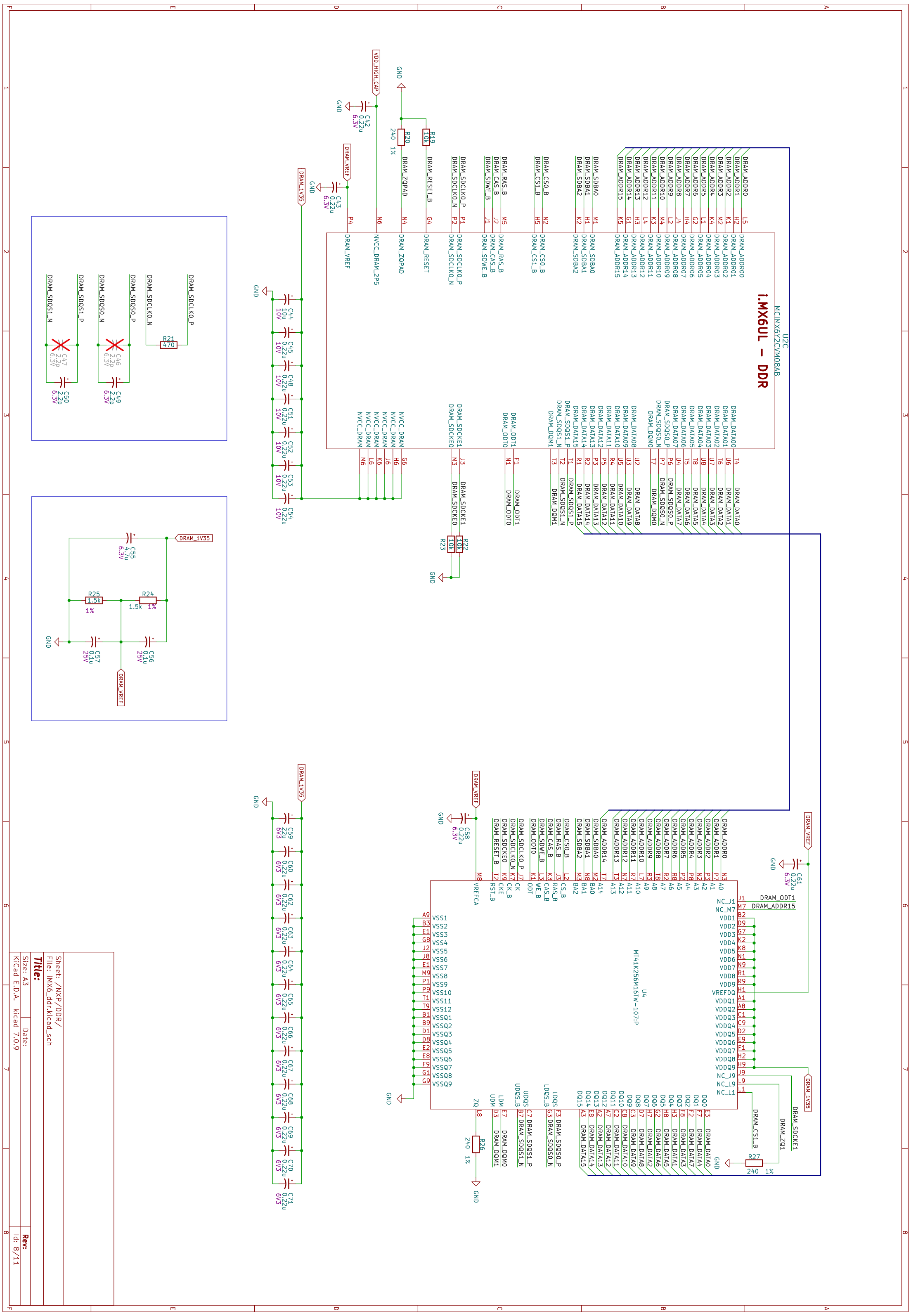


Figura 25: Schema electrică a NXP i.MX 6ULL – DDR

Figura 26: Schema electrică a NXP i.MX 6ULL – Harta memoriei OTP

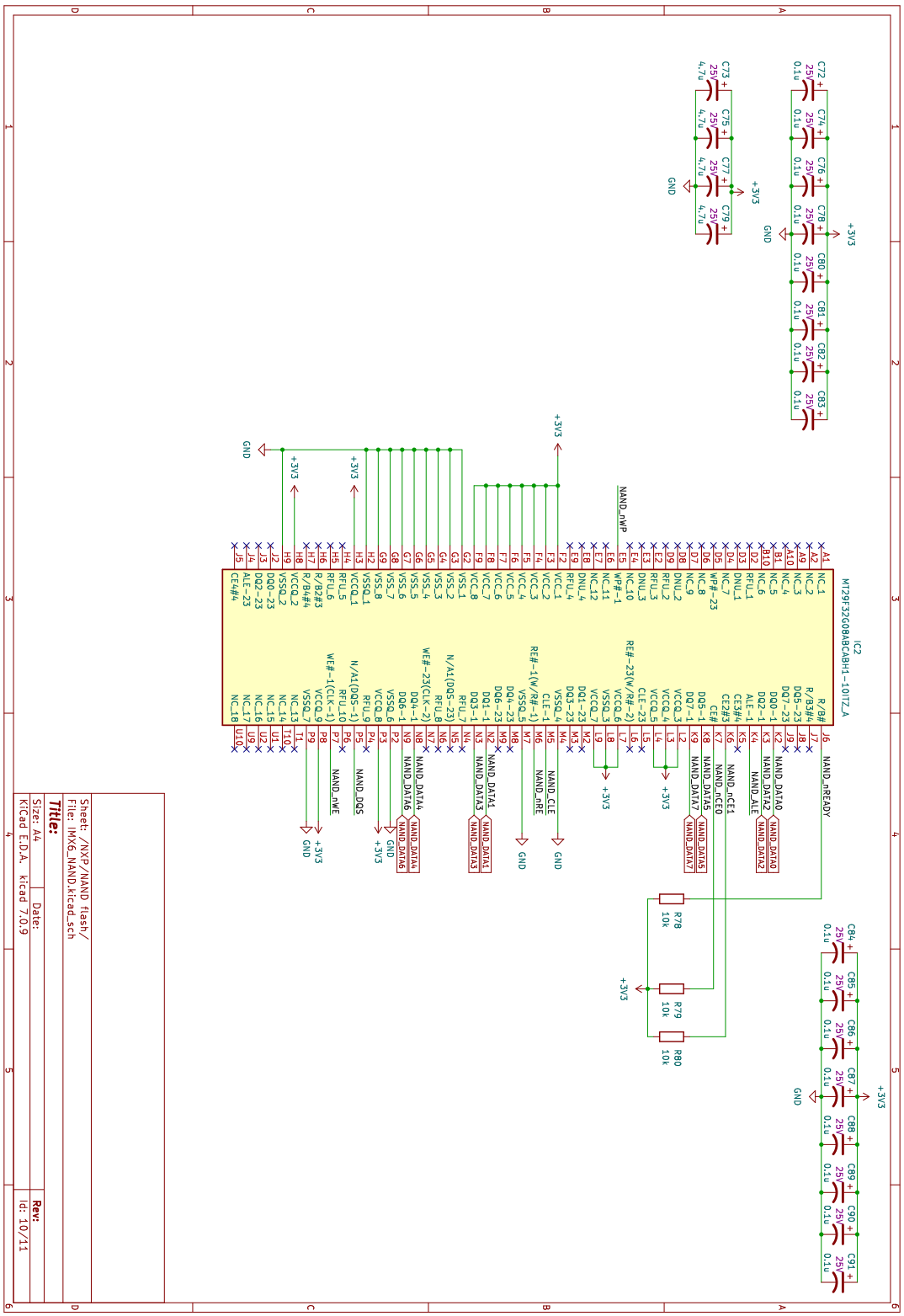


Figura 27: Schema electrică a NXP i.MX 6ULL – NAND Flash

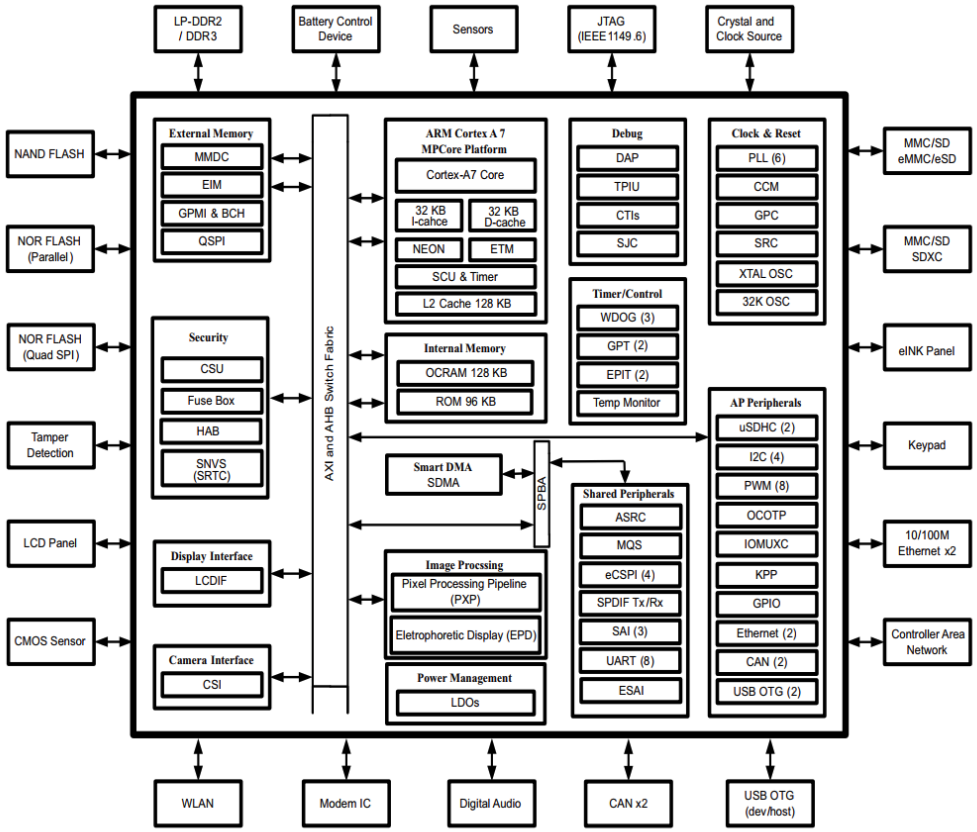


Figura 28: Diagramă bloc – procesor MCIMX6Y2CVM08AB, preluat din "Fișă de catalog: i.MX 6ULL Applications Processors for Industrial Products, NXP Semiconductor, Rev. 1.2, 11/2017".

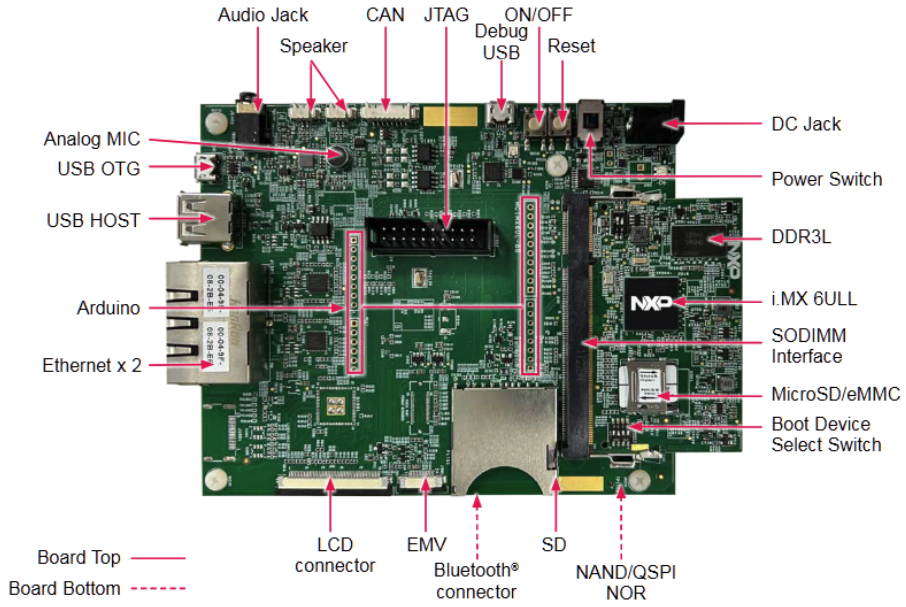


Figura 29: Placă de dezvoltare pentru procesoare din familia MCIMX6ULL, preluat din "Quick Start Guide i.MX. 6ULL Applications Procesor, NXP Semiconductor, IMX6ULLQSG REV 3".

2.8 Publicarea de articole ştiinţifice

Articole de revistă:

J1) **Sperilă, A.**, C. Oară, **B. D. Ciubotaru** și Ş. Sabău. "Distributed control of descriptor networks: A convex procedure for augmented sparsity (accepted; in press - to appear)". În: *IEEE Transactions on Automatic Control*, **2023**. IEEE
IF: **6.5**. **Q1** (Automation & Control Systems - 11/65), **Q1** (Engineering, Electrical & Electronic - 39/275).

J2) Sabău, Ş., **A. Sperilă**, C. Oară și A. Jadbabaie. "Network realization functions for optimal distributed control (accepted; in press - to appear)". În: *IEEE Transactions on Automatic Control*, **2023**. IEEE
IF: **6.5**. **Q1** (Automation & Control Systems - 11/65), **Q1** (Engineering, Electrical & Electronic - 39/275).

J3) Sabău, Ş., **A. Sperilă**, C. Oară și A. Jadbabaie. "Sparse representations of dynamical networks: A coprime factorization approach (provisionally accepted; under second review)". În: *SIAM Journal on Control and Optimization*, **2023**. SIAM
IF: **2.1**. **Q3** (Automation & Control Systems - 39/65), **Q1** (Mathematics, Applied - 49/267).

J4) **Irofti, P.**, L. Romero-Ben, V. Puig și **F. Stoican**. "Learning dictionaries from physical-based interpolation for water network leak localization (accepted; in press - to appear)". În: *IEEE Transactions on Control Systems Technology*, **2023**. IEEE
IF: **5.4**. **Q2** (Automation & Control Systems - 19/65), **Q2** (Engineering, Electrical & Electronic - 74/275).

Articole de conferinţă:

C1) Mihai, Ş. S., **F. Stoican** și **B. D. Ciubotaru**. "Computing the explicit MPC solution using the Hasse diagram of the lifted feasible domain", pag. 1–7. În: *21st European Control Conference (ECC).*, **2023**. IFAC Papersonline.

C2) Nicu, T. G., **F. Stoican** și I. Prodan. "Smooth approximation of polyhedral potential field in NMPC for obstacle avoidance", pag. 1–6. În: *21st European Control Conference (ECC).*, **2023**. IFAC Papersonline.

Director de proiect:
conf.dr.ing. Bogdan D. Ciubotaru

Referințe

[1] https://isdc.gfz-potsdam.de/grace-fo-isdc/grace-fo-gravity-data-and-documentation/.

[2] **Irofti, P.**, L. Romero-Ben, V. Puig și **F. Stoican**. "Learning dictionaries from physical-based interpolation for water network leak localization (accepted; in press - to appear)". În: *IEEE Transactions on Control Systems Technology*, **2023**.
PUBLICAT DE: IEEE
IF: **5.4**. **Q2** (Automation & Control Systems - 19/65), **Q2** (Engineering, Electrical & Electronic - 74/275).

[3] Mihai, Ş. S., **F. Stoican** și **B. D. Ciubotaru**. "Computing the explicit MPC solution using the Hasse diagram of the lifted feasible domain", pag. 1–7. În: *21st European Control Conference (ECC)*. **2023**.
PUBLICAT DE: IFAC Papersonline.

[4] Nicu, T. G., **F. Stoican** și I. Prodan. "Smooth approximation of polyhedral potential field in NMPC for obstacle avoidance", pag. 1–6. În: *21st European Control Conference (ECC)*. **2023**.
PUBLICAT DE: IFAC Papersonline.

[5] Sabău, Ş., **A. Sperilă**, C. Oară și A. Jadbabaie. "Network realization functions for optimal distributed control (accepted; in press - to appear)". În: *IEEE Transactions on Automatic Control*, **2023**.
PUBLICAT DE: IEEE
IF: **6.5**. **Q1** (Automation & Control Systems - 11/65), **Q1** (Engineering, Electrical & Electronic - 39/275).

[6] Sabău, Ş., **A. Sperilă**, C. Oară și A. Jadbabaie. "Sparse representations of dynamical networks: A coprime factorization approach (provisionally accepted; under second review)". În: *SIAM Journal on Control and Optimization*, **2023**.
PUBLICAT DE: SIAM
IF: **2.1**. **Q3** (Automation & Control Systems - 39/65), **Q1** (Mathematics, Applied - 49/267).

[7] **Sperilă, A.**, C. Oară, **B. D. Ciubotaru** și Ş. Sabău. "Distributed control of descriptor networks: A convex procedure for augmented sparsity (accepted; in press - to appear)". În: *IEEE Transactions on Automatic Control*, **2023**.
PUBLICAT DE: IEEE
IF: **6.5**. **Q1** (Automation & Control Systems - 11/65), **Q1** (Engineering, Electrical & Electronic - 39/275).

[8] Andersson, J. A., J. Gillis, G. Horn, J. B. Rawlings și M. Diehl. "CasADi: a software framework for nonlinear optimization and optimal control". În: *Mathematical Programming Computation*, pag. 1–36, **2019**.
PUBLICAT DE: Springer.

[9] Althoff, M. și N. Kochdumper. "CORA 2016 manual". În: *TU Munich*, **2016**.
PUBLICAT DE: Citeseer.

[10] Simon, D. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*, **2006**.
PUBLICAT DE: John Wiley & Sons.